SIGS See Us at XML ONE Fall 99, November 8–11, 1999 see p.71

Java™ REPORT

October 1999
Volume 4, Number 10
$4.95 US/$7 Can/$4.95 Int'l

THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

It's Hot... It's Portable...

XML

Using XML As a Distributed Computing Protocol

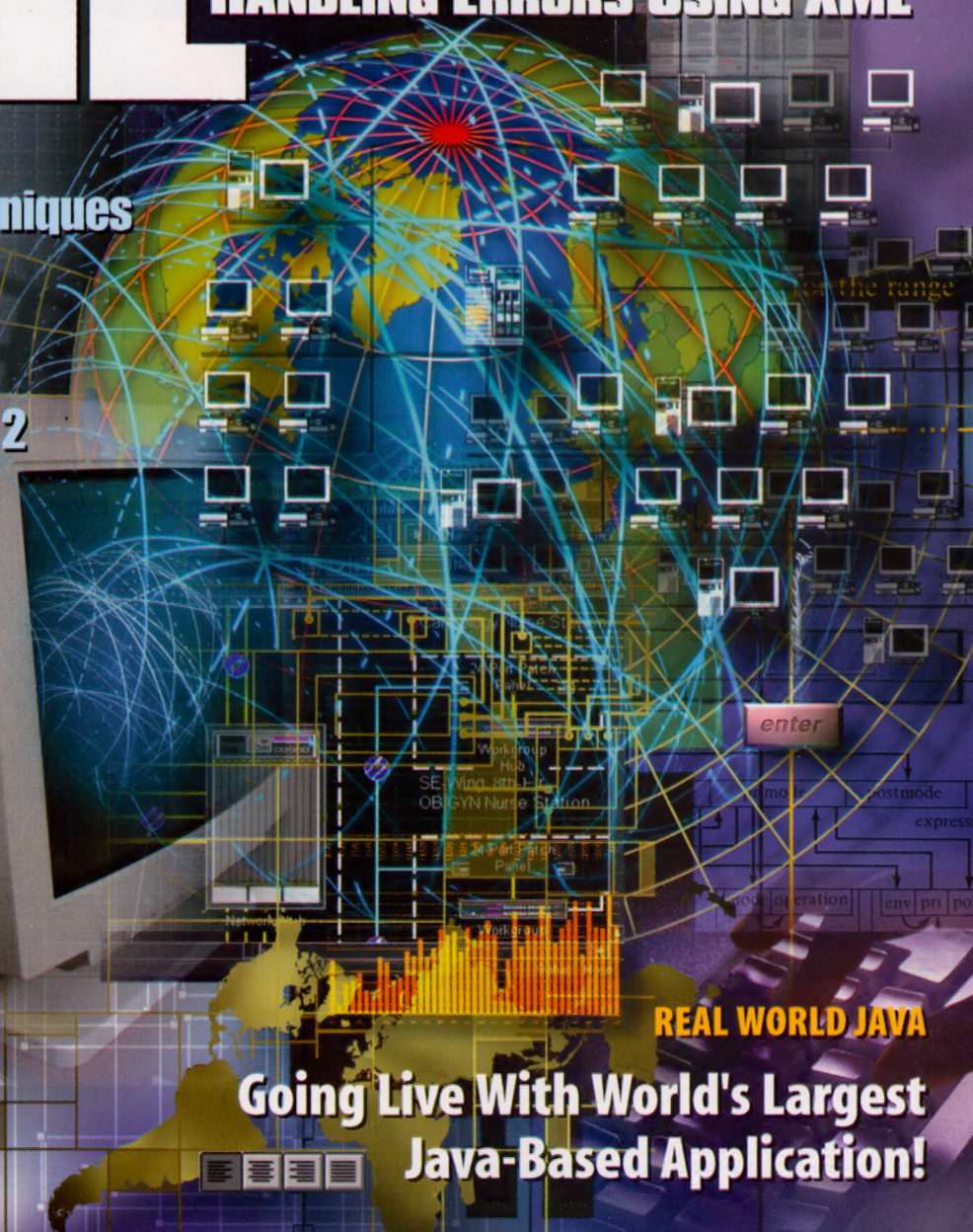
Handling Errors Using XML

Instance Buffering Techniques Using Client-Side GUI Component Instances

Details Behind the Java 2 RMI Architecture

CORBA Component Model Specs Soon to Be Ratified by OMG

Novel Ways to Use the Reflection API



REAL WORLD JAVA

Going Live With World's Largest Java-Based Application!

0 74470 88941 0 10

To Get to Market Faster

Get a Firm Grip on Your e-Security.

RSA BSAFE® security components give you the control you need to beat the competition.

Developers, start your applications. The race to e-business is on. Whether you're developing apps for web business, general client-server, remote management or cell phones, one thing is clear: You need to get a firm grip on e-security or face a big potential pile-up.

Never fear, the RSA BSAFE pit crew is here. To maximize your opportunities for success in e-business, you need maximum security from the industry leader, RSA Security. Our RSA BSAFE security components give you all the tools you need to rev up SSL in C, SSL in Java™ and S/MIME implementation way beyond what street-issue browsers can offer. Such as full strength crypto available worldwide,* more extensive sample code and protocol links for services such as Telnet, FTP, NNTP, SMTP and POP3.

By combining SSL and S/MIME security protocols with the industry's most trusted cryptography, RSA BSAFE components give you the ability to negotiate the twists and turns of security protocol implementation in record time. And come out safely in the lead.

All our tools build confidence. Maneuverability and creative thinking will help you win the race, and with RSA BSAFE components you have the power to choose how our air-tight e-security works best within your application. You control the interface, error recovery and the best balance of strength vs. accessibility for your needs.

Best of all, you run with the confidence that your e-security operates independently of changes to the browser. No yellow caution flags for separate browser versions, either. With RSA BSAFE components, your apps will be completely interoperable.

Welcome to the winner's circle. It takes a winning team to win the race. RSA Data Security has been a driving force behind putting SSL and S/MIME on the map as industry standard security protocols. Today we're known as RSA Security, and no one knows more about how to build secure applications. So you can count on us to provide tools you can use to meet your goals on time and on budget.

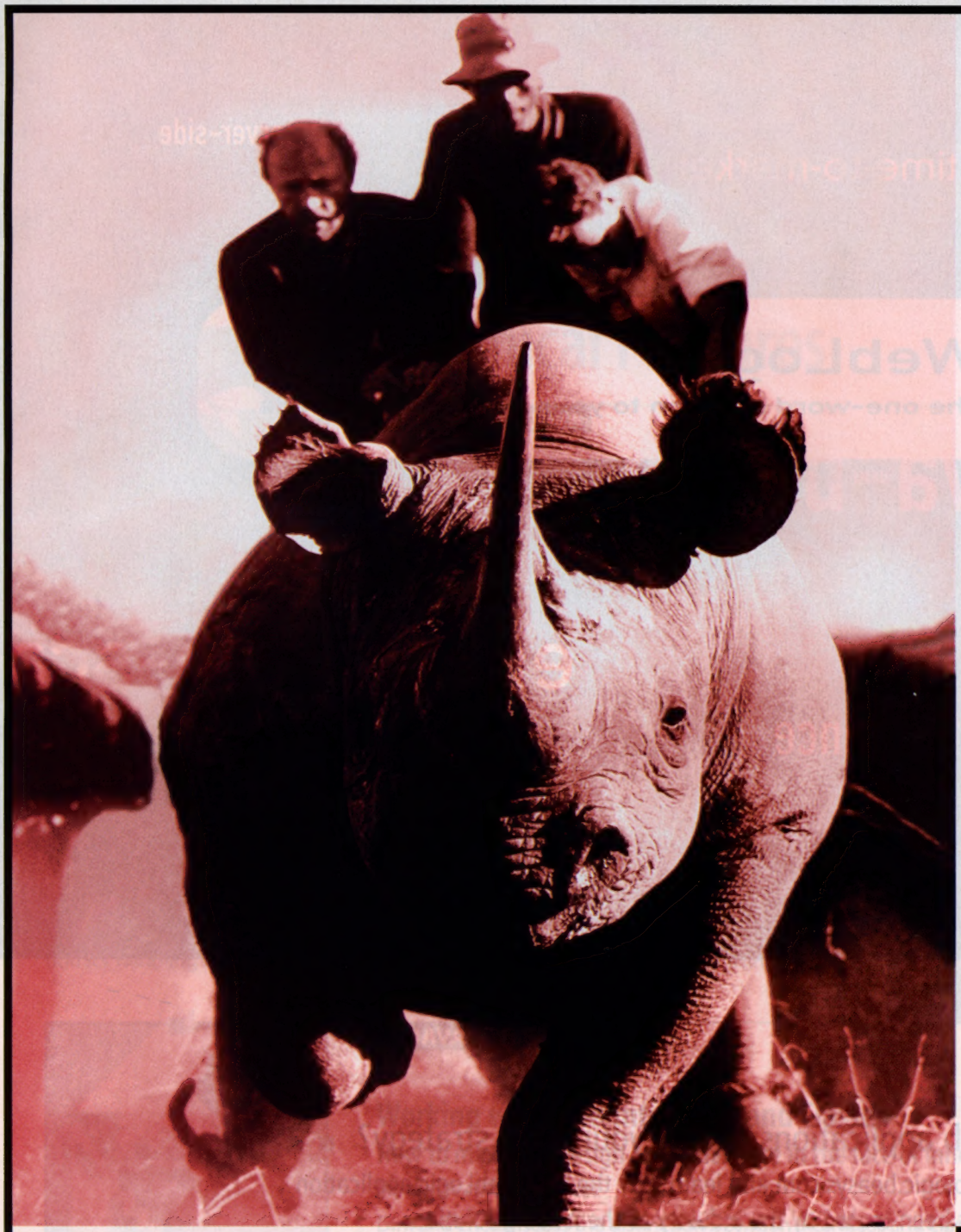
It's never too late to get a jump start on the competition – especially when the stakes are so high. So don't waste another minute idling. **Qualify today for a free RSA BSAFE SSL-C, SSL-J or S/MIME-C software development kit* and head to the starting line.**

1.800.782.5453 or www.rsasecurity.com/win

RSA SECURITY™

The Most Trusted Name in e-Security

*Export or other restrictions may apply. RSA, BSAFE and the RSA Security logo are trademarks or registered trademarks of RSA Security Inc. Java is a trademark of Sun Microsystems, Inc. © 1999 RSA Security Inc. All rights reserved.



UNTIL NOW, JAVA FOR EMBEDDED SYSTEMS HAS BEEN JUST TOO BIG AND UNPREDICTABLE.

INSIGNIA SOLUTIONS

JEODE™

You want the power of Java™. But for your embedded application, it's just too hefty, unpredictable, and not fast enough. Insignia Solutions® introduces Jeode™, our implementation of Java, which is fully compliant with Sun's EmbeddedJava™ specification and uniquely designed for embedded systems. Jeode lets you optimize size, performance and predictability for your specific embedded application. Featuring our proprietary EVM™ (Embedded Virtual Machine™), Jeode is highly configurable and tunable to achieve optimal performance in resource-constrained embedded systems. The EVM provides adaptive optimizing dynamic compilation for fast execution and concurrent garbage collection for predictable behavior. With more than a decade of delivering virtual machines, Insignia Solutions is already providing Jeode to Fortune 500 makers of networking equipment, PDAs, mass storage systems, consumer electronic devices and more. So, can Java be small enough, fast enough and predictable enough for your embedded system? The answer is a resounding yes! With Jeode. Call **1.800.848.7677** or visit **www.insignia.com**.

Code JR

Insignia Solutions is a registered trademark and Jene, EVM, and Embedded Virtual Machine are trademarks of Insignia Solutions, Inc. Sun, Java, and EmbeddedJava are trademarks or registered trademarks of Sun Microsystems, Inc. All other trademarks are the property of their respective holders.

ease-of-use

leading-edge

server-side

time-to-market

WebLogic.

The one-word solution to your hyphenated needs.

java-based

mission-critical

best-of-breed

enterprise-sca

high-performance

time-to-market

Are Web applications part of your 1999 strategy?

Go to weblogic.beasys.com and learn why the BEA WebLogic application server has been touted by the industry as the leading solution for quickly deploying and managing your high-performance, mission-critical, Java-based, enterprise-scale (etc.) Web applications.

Crossroads A-List Award:
"Rapid Application Development Tools for Java" (January 1999)
JavaWorld Editor's Choice Award:
"Best Web Application Server" (December 1998)
Java Developer's Journal Editor's Choice Award:
"Best Web Application Server" (December 1998)
Intelligent Enterprise RealWare Award:
"Best Application Using a Component Architecture" (December 1998)

BEA WebLogic®

weblogic.beasys.com

© 1999 BEA Systems, Inc. All rights reserved. BEA, the BEA logo, and WebLogic are registered trademarks of BEA Systems, Inc. Java is a trademark or registered trademark of Sun Microsystems, Inc. Other trademarks are the property of their respective owners.

BEA

October 1999
Vol. 4, No. 10

Java™ REPORT

THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

Technical Features

ENTERPRISE JAVA

XML as a Distributed Application Protocol 16

The Technology and the Politics

Richard Deadman

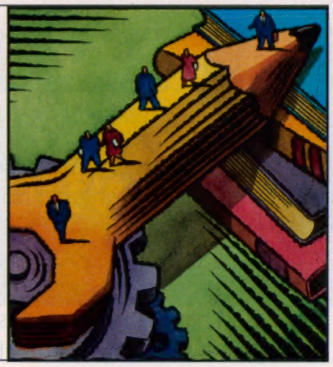
It's the latest buzzword in the Internet application arena: XML. It's hot. It's powerful. Everyone loves it. But isn't this exactly what CORBA already provides? Richard explores why XML is so hot, shows how XML can be used as a distributed computing protocol, and looks at its advantages, disadvantages, and appropriate uses.

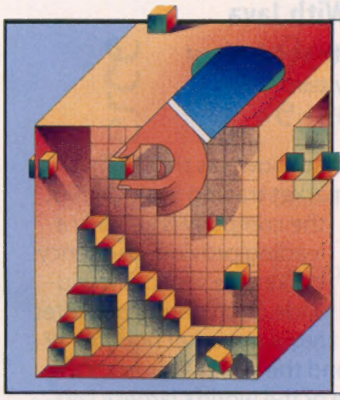
POWER JAVA

Error Handling with XML 22

Steven Jones

Error handling is important for any serious application and can be especially tricky for distributed applications. RPC Middleware platforms such as CORBA, RMI, and EJB provide exceptions for reporting errors across process boundaries—this can go a long way toward a comprehensive error handling approach. Steven presents a unique solution for handling errors in a language and a platform-independent way of using XML.





JAVA PRIMER

Working with Java Interfaces and Classes 34

How to maximize design and code reuse in the face of inheritance

Dirk Riehle and Erica Dubach

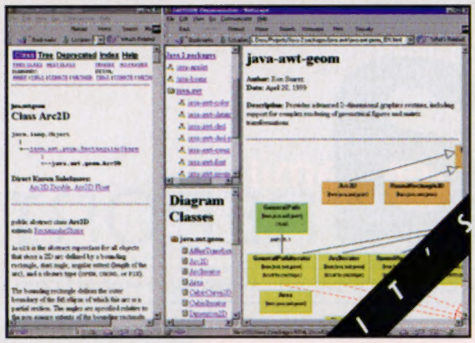
Dirk and Erica discuss the distinction between interfaces and classes as interface implementations and describe how to deal with interface inheritance and how to factor implementations to maximize code reuse. This is the second in a series of two articles.

POWER JAVA

Flexible Strategies for Optimizing Application Performance 48

Christopher A. Fregly

As application complexity and functionality increase, application performance decreases proportionally. The simple, short-term solution for improving application performance is to upgrade the hardware involved. The better, long-term solution is to optimize the application design and implementation. By maximizing the use of all available resources, the application can more easily adapt to changes in deployment environment, complexity, and functionality.



Check out our new interactive version of the Java 2 Core Class Libraries at www.javareport.com/java2interactive

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com
October 1999 Vol. 4, No. 10

Columns

DISTRIBUTED COMPUTING 60

Remote object activation

Govind Seshadri

Govind examines the details behind a significant enhancement to the RMI architecture under the Java 2 platform.

BUSINESS OBJECTS 70

CORBA components—alive and well

David S. Frankel

Contrary to rumors, the OMG CORBA Component Model is very much alive. While it is true that the CCM architects have gone to great lengths to ensure that CCM plays well with EJB, such characterizations are overly simplistic. CCM has features besides language-neutrality that go considerably beyond EJB.

SCOTT'S SOLUTIONS 80

Using the Reflection API

Scott Oaks

Scott discusses some novel ways of using the Reflection API. The starting point for his exploration is a solution first shown to him by Mike D'Orazio and Ray Washburn at AT&T.

Check out Java Report Online!

- *Selected features and columns from the print edition*
- *All code listings are available online!*
- *Monthly product updates*
- *Java Report Archives*

Departments

EDITOR'S NOTE 6

So, What's XML?

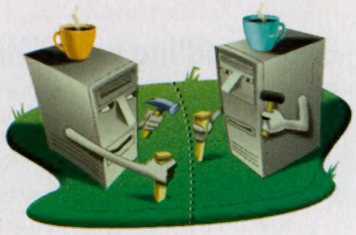
Dwight Deugo

LETTERS TO THE EDITOR 8

BUZZ 10

Philip J. Gill

The First Java ASPs Stake Out Their Turf; Sun Dubs Project StoreX "Jiro," Shows Spec; Beware of Killer Apps for Non-PC Devices; and much more.



REAL WORLD JAVA 12

Trading in Real Time With Java

Instinet to go live with one of the world's largest real-time Java-based trading systems

Philip J. Gill

Instinet Corp. is perhaps best known as the leading provider of "after-hours" market trading in stocks. But the company is actually a leading provider of agency brokerage services worldwide, both during and after regular market hour, for 17 securities exchanges around the world. This fall, Instinet goes live with one of the world's largest Java-based applications. It will serve thousands of customers with real-time brokerage services in the buying and selling of fixed income instruments.

INSTINET® A REUTERS Company

AD INDEX 64

Cover image by Jeff Brice

TEST DRIVE

Persistence PowerTier 5 for EJB

Download FREE Evaluation Software!

www.persistence.com/Download.html Priority Code JR10

The Engine for E-Commerce

Developing high-volume, high performance e-commerce applications?

Download a free evaluation copy of PowerTier™ 5 for Enterprise JavaBeans™ and discover how Persistence brings your e-commerce system to the next level of performance, scalability and personalization!

PowerTier 5 for EJB

- *Delivers application performance through its in-memory cache*
- *Ensures scalability and high availability with PowerSync technology*
- *Reduces time-to-market with patented O-R mapping*

Persistence

The Engine for E-Commerce

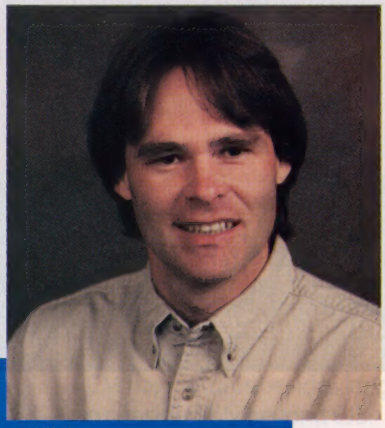
800.803.8491 • info@persistence.com • www.persistence.com

© 1999 Persistence Software, Inc. All Rights Reserved. Persistence is a registered trademark and PowerTier is a trademark of Persistence Software in the United States and other countries. U.S. Patent Nos. 5,499,371; 5,615,362; 5,706,506

So, What's XML?

IT HASN'T BEEN very long since Sun began its efforts to define a Java standard extension for the Extensible Markup Language (XML). However, Sun has moved quickly to bring developers access to XML documents from Java. On May 9, 1999, it announced its intent to create a Java platform standard extension for XML through the Java Community Process. The review period for its XML Data Binding Java Specification Request (JSR-000031) closed on August 23rd. And, for several months now you could download a complete set of tools for developing full-scale XML-oriented applications and network services from several companies, including Sun's Java Project X Technology, Release 2.

So what's XML? In appearance, it is similar to HTML, but it gives you more than just the ability to format documents. With XML you can identify the kinds of information in the document. Therefore, you can easily reformat the information, and, more importantly, you can process it. For example, I have been looking at using XML to structure software patterns. I am not attempting to develop one definitive format for patterns, as I don't think one exists. Rather, I want to be able to support all formats. Using XML, I can determine the format of a pattern; I can verify a pattern adheres to the structure of the format; and, I can reason about the information a pattern contains because I know where its parts are located. With this information, my belief is that I will be able to develop better search tools for patterns and be able to link related ones together in new ways. What better way to do this than by building the tools in Java? In the words of Jon Bosak, known as the father of XML technology, XML and Java technology are "the yin and yang of cross-platform XML technology." XML gives you data portability and Java gives you platform portability.



Dwight Deugo
deugo@scs.carleton.ca

In addition to data processing, where XML encodes the data for a program to process, you can use XML in other ways. You could use XML for document-driven programming, where references to user interface components and business-logic components are combined in a document and then used to create an application on the fly. You could archive components in XML for resurrection in the future, but with the added abilities of easy transport and modification. You could do...well you figure it out. The potential for you is there. If there ever was a perfect marriage, Java and XML would be it, and they would even let you join in too!

Dwight Deugo

Java™ Report (ISSN 1086-4660) is published monthly by 101communications, LLC, 71 West 23rd St., 7th Floor, New York, NY 10010, 212.242.7447. © Copyright 1999 by 101communications, All rights reserved. Periodicals postage paid at New York, NY, and additional mailing offices. Regular subscription rates— $39 domestic, $64 Canada and Mexico, $88 foreign; institutional—$139 domestic, $164 Canada and Mexico, $188 foreign. Postmaster: Send address changes and subscription orders to: Java Report, P.O. Box 5050, Brentwood, TN 37024-5050. GST (Canada) #R895884229. For service on current subscriptions: in North America call 800.361.1279 (v) or 615.370.4845 (f); (custserv@cwixmail.com (email); outside North America +44(0)1858 435 302 (v), +44(0) 1858 434 958 (f); custserv@cwixmail.com (email). Statements of opinion and fact are made on the responsibility of the authors alone and do not imply an opinion on the part of 101communications, or the editorial staff. All trademarks are the property of their respective owners. Printed in the USA.
Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries. *Java™ Report* is independent of Sun Microsystems, Inc.

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT

Editor-in-Chief
Dwight Deugo; *deugo@scs.carleton.ca*

Executive Editor/Managing Editor
Kathleen M. Major; *kmajor@sigs.com*

Art Director
Jan Foster; *jfoster@sigs.com*

Assistant Managing Editor/ Associate Online Editor
Dan Olawski; *dolawski@sigs.com*

Assistant Editor
Melissa Shepherd; *mshepherd@sigs.com*

Associate Art Director
Fernando Quiñones; *fquinones@sigs.com*

Associate Editor
Philip J. Gill; *philipgill@aol.com*

Product Editor
John K. Waters; *john@watersworks.com*

COLUMNISTS
Steve Ball, *Independent Consultant*
John Crupi, *Sun Microsystems*
Mark Davis, *IBM*
Steven Disbrow, *EGO Systems*
David S. Frankel, *Genesis Development Corp.*
Vincent Hardy, *Sun Microsystems*
David Jordan, *Ericsson*
Lowell Kaplan, *Random Walk Computing*
Craig Larman, *Object Space*
John Miller Crawford, *Dave Clark Design Associates*
Scott Oaks, *Sun Microsystems*
Govind Seshadri, *Strategic Technology Resources*
Tig Tillinghast, *Internet Profiles*

101communications SIGS PUBLICATIONS

101communications, LLC
Curt Hessler, *Chairman and CEO*
Jeffrey S. Klein, *President and COO*
Michael Weaver, Senior *Vice President and CFO*
Christine Oldenbrook, *Vice President/Circulation*
Richard P. Friedman, *President, SIGS Conferences/ SIGS Europe*
Charles P. Shively, *Group Publisher*

Corporate Headquarters
5900 Sepulveda Blvd., Suite 560, Van Nuys, CA 91411
v. 818.756.0566 • www.101communications.com

For subscription customer service please call toll-free 800.361.1279

ADVISORY BOARD
François Bancilhon, *Ardent Technology*
Grady Booch, *Rational Software Corp.*
Alan Brown, *Sterling Software*
Thomas Keffer, *Rogue Wave Software*
Bertrand Meyer, *ISE*
Meilir Page-Jones, *Wayland Systems*
Jan Popkin, *Popkin Software and Systems, Inc.*
Kenneth S. Rubin, *Salient*

Publishers of *JOOP, C++ Report, Java™ Report, Application Development Trends, OBJEKTSpektrum (Germany), Java™SPEKTRUM (Germany), Application Development Advisor (UK), Java™ Report Online, C++ Report Online,* and *JOOP Online.*

500,000 downloads of Sun Microsystems' Java™ Foundation Classes.

Multiple platforms.

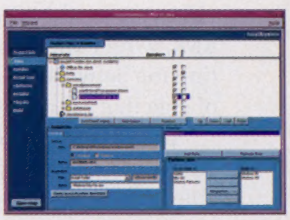
One installer.

InstallAnywhere™

The Most Powerful Multi-Platform Deployment Solution

ZERO G SOFTWARE

You've invested countless hours developing your software. The last thing you want to worry about is deploying it to the various operating systems your customers are using. With InstallAnywhere, you can rest easy: InstallAnywhere is the only installer that handles all the platform specific details for you, whether it's custom icons on Mac, registry entries on Windows, or shell scripts on Unix. InstallAnywhere supports virtually every Java technology-enabled platform, allowing you to create installers with a professional look and feel that's familiar to all users on all platforms. When you want your software to run everywhere, think InstallAnywhere.



www.ZeroG.com

Zero G Software and InstallAnywhere are registered trademarks of Zero G Software, Inc. Sun, Sun Microsystems, Java, Java Foundation Classes, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All other trademarks and registered trademarks are the property of their respective holders.

Resource Bundle Update

This is in response to Paul Copeland's letter to the editor (*Java Report*, June 1999, Vol. 4, No. 5) regarding my article, "Using Resource Bundles to Internationalize Text" (*Java Report*, February 1999, Vol. 4, No. 2). I have sent the same to Paul.

Thanks for giving users more information on Resource Bundles. I have been meaning to write an update to that article for some time; however, new articles have taken priority.

The lower-level method was used and discussed because the work and the article were done before the Java Plugin was available, giving a common JVM. The application needed to support resource bundles on NT, 95, and OS2 using both MS IE and Netscape. There were several limitations using `getBundle` with the browser to support JVM, (security issues on the different browsers on the different platforms), at least at that time. Also, in my opinion, having to implement different classes for the different resources, especially when the classes were basically the same, was a waste. I have since enhanced the actual code to use `getBundle` (or the low-level method if that fails) without having to implement all the classes required by the default approach.

Again, thanks for your response. The goal of the article was to inform, and together we have potentially helped a number of developers.

Bob Patten
wrpatten@us.ibm.com

Fact or Fiction?

Lowell Kaplan's July 1999 column, "Distributed computing in Java: RMI, CORBA, or both?" (*Java Report*, Vol. 4, No. 7) describes RMI remoting incorrectly and attributes fictional benefits to it. The third of his four differences between RMI and CORBA, "RMI passes objects by value; CORBA passes objects by reference," misstates the case. In discussing this point, he correctly describes how CORBA employs proxies to make communication with remote objects transparent. But in saying "RMI delivers not just a reference to a remote object, but an actual copy of the object, with all of its data and methods intact," he simply got it wrong.

RMI remote references differ from CORBA ones in many subtle ways, but in this fundamental respect they behave identically. Both employ proxies to transmit method requests to the remote object and return the responses, nothing more. Lowell's consequent argument that RMI offers performance benefits because it copies the entire remote object to the client, simply has no basis in fact. I doubt this capability "has been one of the biggest reasons for many architects' decision to use RMI." If it has, then I foresee career changes for those architects. I believe that anyone familiar with the RMI spec, or having modest first-hand experience programming with RMI, would know better. RMI and CORBA both make a distinction between a first-class, remote object and other kinds of data that get passed by value.

In CORBA, simple types, structs, unions, and any objects get passed by value. In RMI, primitive types and serializable objects get passed by value. The RMI spec uses the term pass by *copy*; it clearly distinguishes between these serializable objects, which it calls nonremote objects, and remote objects (p. 10).

The designers of RMI–IIOP set out to make CORBA accessible to programmers accustomed to the RMI programming style. Before the objects-by-value spec, CORBA had no way to represent the equivalent of a serializable Java object that you could pass, by copy, through a remote interface (i.e., over the network) and then operate on in your local address space. The CORBA struct wouldn't do, because it did not have methods. The OMG had to add the notion of value objects to CORBA, a major initiative. RMI–IIOP essentially defines a new style of CORBA programming that clearly acknowledges the primacy of Java but also maps into other languages. When ORB vendors ship C++ (or other language) implementations of objects by value, RMI–IIOP could become a multi-language standard for interoperability. Java RMI programmers can use the RMI–IIOP programming model today to keep that option open for the future, without compromising the benefits of RMI.

Owen Tallman
owen.tallman@acm.org

RMI vs. CORBA for Java IDL?

I have two comments regarding Lowell Kaplan's July (*Java Report*, Vol. 4, No. 7) column, "Distributed computing in Java: RMI, CORBA, or both?"

First, the author apparently assumes that passing objects by value is always more efficient than passing objects by reference. But this is not the case. It depends on the number of invocations of methods on the object and their parameters on the one side, and the amount of bytes, which have to be transferred by passing the object, on the other side.

Second, the author assumes that classes coded as RMI can always work with CORBA classes, if RMI over IIOP is used. But this is not the case. IIOP is the language, but this does not make sure that "the one class says, what the other wants to hear." See the FAQs at **www.ibm.com**, especially for the answer to the question: Should I be using RMI over IIOP for JavaIDL?

Hermann Schmitt

Search Engine for *Java Report Online*

I highly appreciate the content in your publication and regularly circulate your articles to my coworkers at group meetings. I encourage them to subscribe or buy your magazine, but I have been confronted with the impossibility of researching your past articles online. This is a cause of frustration because I am forced to settle for articles of less value in other online publications that do have a search engine. If there is a search engine for your Web site, then please relocate it to a more visible position so that it can be found easily. On the other hand, if the search engine is not available, then I hope that it will be implemented soon. We, the Java development community, would highly appreciate it.

Hugo A. Garcia
hugo_garcia@standardandpoors.com

We have been testing search engines and should have one available online in the near future. We are always researching ways to make the online information easier to access.

The cumulative author index is available online now and provides a list of every Java Report *article published from the first issue in 1996 up to the December 1998 issue.*

Dan Olawski
Associate Online Editor

continued on page 32

It was the day Speiros™ was introduced and every computing resource on the planet became available to every person on the planet. That's because Speiros technology makes it possible for disparate computing technologies to converge into a powerful Internet-based platform of interoperability. It may have seemed like any other day, but it was the beginning of true pervasive computing.

To learn more, visit www.cyrusintersoft.com or call 1-877-88CYRUS.

cyrus™ INTERSOFT, INC.

ENABLING CHOICE™

Sun Dubs Project StoreX "Jiro," Shows Spec

Sun Microsystems, Inc. (www.sun.com) has given the all-Java storage platform initiative, code-named "Project StoreX," a formal name, the Jiro platform.

Jiro, implemented in both Java and Jini technologies, is an open storage management platform that will enable customers to manage mixed environments of servers, storage, and devices on a network, Sun said.

"The Jiro platform provides the common denominator for developers of storage management applications and will drive the creation of more flexible and efficient storage management solutions," said **Denise Shiffman**, Sun's director, content services. "The Jiro platform will be the glue that [binds] various storage frameworks and applications together as well as [that simplifies] infrastructure developers need to work with to create interoperability."

In addition, Sun said the Jiro platform expert group, formed in April 1999, has released the draft Jiro specification for participant review. To view the draft specification currently in the revision process, visit **www.jiro.com**.

Be Repositions OS for Devices

Be Inc. (www.be.com) apparently hopes it can find the fame and fortune that has thus far eluded it in the mainstream desktop computer market, in the emerging market for Internet appliances.

Of course, in that market it will be facing not just Microsoft, with its WindowsCE operating system (OS), but a long list of real-time embedded OS vendors who have longer histories and better stories to tell. These include **Symbian Corp.'s (www.symbian.com)** Symbian, **WindRiver Systems Corp.'s (www.windriver.com)** Vrx, **Geoworks Inc.'s (www.geoworks.com)** GEOS, **QNX Software System Ltd.'s (www.qnx.com)** QNX , **Microware Corp.'s (www.microware.com)** OS9, and Linux, an open source Unix look-alike available in more than a dozen commercial distributions. All of them, by the way, are supporting Java, in one form or another, as the primary application development language for Internet appliances using their OSes.

continued on opposite page ➤➤

The First Java ASPs Stake Out Their Turf

By the time you read this, the first two Java application server providers (ASPs) should be off and running, staking claim to a market estimated to reach $2 billion by 2003, according to market research house **International Data Corp. (IDC;www.idc.com)**, Framingham, MA.

A second market research firm, **Forrester Research Inc. (www.forrester.com)** of Cambridge, MA, has much bigger expectations. According to Forrester, the ASP market will be a $6 billion business by 2001, for the most part catching on with small to mid-size companies.

The two Java entries are startup companies—**AgencyWorks Inc. (www.agencyworks.com)** and **Kairos Damango Inc. (www.damango.com)**. They are jumping into a fray now dominated by much larger enterprise resource planning (ERP) software vendors, including **SAP America Inc. (www.sap.com)**, Newtown Square, PA; **Oracle Corp. (www.oracle.com)**, Redwood Shore, CA; and **PeopleSoft Inc. (www.peoplesoft.com)**, Pleasanton, CA; as well as a pair of dedicated year-old ASP startups, **Corio Corp. (www.corio.com)** of Redwood City, CA and **USinternetworking Inc. (www.usi.net)** of Annapolis, MD.

But rather than going after corporate customers with ERP applications or e-commerce application hosting like their larger rivals, the two Java startups are concentrating on market niches in which they expect Java's benefits of "write once, run anywhere" will hit home.

The first, AgencyWorks, based in Salt Lake City, went live in early August and is targeting the insurance agency market; specifically, broker agencies that wholesale various insurance products from the big carriers to independent sales agencies nationwide. It will first target the individual life insurance market, then later plans to address the property and casualty market, said **Greg Reynolds**, the company's executive vice president.

All of AgencyWorks' applications are written in a combination of Java, for the presentation and client side, and RPG, for the AS/400 server that hosts them. Their first two products are the Hosting Engine, containing business logic and an online database of life insurance products, underwriting guidelines, and quoting guidelines; and Agency Integrator, the all-Java client-side desktop.

The Java application and presentation layers were written using Jacada, a Java integrated development environment (IDE) from **Computer Systems Technology Corp. (CST; www.cst.com)**, Atlanta. The Hosting Engine

Sun–Netscape Alliance Ships First Products

The Sun–Netscape Alliance has shipped its first product, the Netscape Application Server 4.0. The Alliance also announced the Netscape Application Builder 4.0 development environment for building enterprise-class applications.

New features in the Netscape Application Server 4.0 include: support for the Enterprise JavaBeans (EJBs) software component model; integration of the Java Server Pages (JSPs), JDBC, and Java Servlet API technologies; support for distributed transactions through tight integration with IBM TX Series; improved performance; and a new self-tuning load balance option.

Netscape Application Server 4.0 is priced at $35,000 per CPU. Netscape Application Builder 4.0 is priced at $1,295 per developer seat.

The Sun–Netscape Alliance resulted from the acquisition of **Netscape Communications Corp. (www.netscape.com)** by **America Online Inc. (AOL; www.aol.com)**. As part of that agreement, **Sun Microsystems Inc. (www.sun.com)** agreed to form the Sun–Netscape Alliance to co-develop and market Netscape software products.

Illustration by Ned Shaw for Java Report

runs on an IBM AS/400 under the OS/400 operating system and stores data in an IBM DB2 database. All the database operations are written in RPG, the native programming language of the AS/400 family.

Reynolds said future products include Broker Gear, a hosted Web site product offering that allows broker agencies to create a Web presence without building or maintaining one; and Agency Gear, a hosted Intranet for insurance agencies. These applications will also be written in a Java and RPG combination.

According to Reynolds, the ASP approach was the only way to go for the insurance agency market. "Historically, insurance agency software has been subsidized or given away," he said. "Because of that, nobody would buy this software. The ASP approach is the only one that works for this industry."

The second all-Java ASP, from Kairos Damango, will go by the name of Damango, a made-up name that doesn't mean anything, according to **Grant Woods**, the company's technology director. "We want to create our own meaning," he adds.

A subsidiary of Ulysses Telemedia Networks Inc. of Minneapolis, Kairos Damango plans to target the general public with its ASP service, rather than corporate accounts or vertical markets. Woods said the service will deliver on the original vision of network computing, picking up where others failed "because earlier companies in network computing space really had a hardware mission."

Kairos Damango won't be in the hardware business. Instead, the company is planning to offer an online catalog of more than 100 all-Java applications. Users will sign onto their accounts, search the catalog, and click on an application they want to use. Signing up for an account will likely be free, but users will rent the applications of their choice. They'll also be given a disk space.

Because all the applications are written in Java, users will be able to download runtime versions to any hardware platform. It also means users will be able to access their applications and files from any Internet-connected device, such as their desktop at work or home, or a mobile laptop on the road. Woods describes the company's philosophy as "write one, run everywhere."

The underlying technology for Kairos Damango's ASP is Speiros 1.0, an all-Java network operating system that makes Java applications available to any user, anywhere, anytime. Speiros was developed by **Cyrus Intersoft Inc. (www.cyrusintersoft.com)**, also of Minneapolis. Cyrus, by the way, was spun off from the same parent company last year as Kairos Damango. They previewed a beta version of Speiros at this year's JavaOne, and will introduce a production version this month at Fall Internet World '99 in New York. (See the following buzz item.)

Beware of Killer Apps for Non-PC Devices

In a new research report, **Forrester Research Corp. (www.forrester.com)** analyst **Paul Sonderegge** warns that companies attempting to serve all non-PC devices—including Web TVs, personal digital assistants (PDAs), and cell phones—will fail. Instead, firms must target select devices for which custom-coded services will benefit consumers most.

Cyrus Intersoft to Deliver Speiros 1.0

After making an initial splash at this year's JavaOne conference in June with a preview beta release of Speiros, a Java-based network computing operating system (OS), Minneapolis, startup **Cyrus Intersoft Inc. (www.cyrusintersoft.com)** said it's ready to ship a production version, Speiros 1.0, this month.

Speiros 1.0 will formally debut at Fall Internet/World '99 in New York early this month, with the first commercial shipments slated to go out the door within a week thereafter, according to a company spokeswoman.

Cyrus Intersoft director of engineering Kevin Herrboldt told *Java Report* the company has made several enhancements to the beta that will appear in the 1.0 production release. Those features include the ability to save files to a local hard disk, as well as to a network server or hard disk, a customizable user desktop, and a catalog system and catalog browser that will allow users to search for available applications and launch them by clicking on an icon.

Speiros is an all-Java network computing OS that makes Java applications available across the network from any server to any client device.

➤➤ *continued from previous page*

Be does have one win thus far, **Microworkz Inc.'s (www.microworkz.com)** iToaster, a low-cost PC-like device for Internet access. That apparently has given the company hope.

IBM Adds to developerWorks Site

IBM (www.ibm.com) has enhanced its developerWorks Web site at **www.ibm.com/developerWorks** with new sections on Web architecture and Linux, the popular open source Unix clone. It's also expanding the content and resources of a new Unicode section.

BEA Buys Java Training Co.

BEA Systems Inc. (www.beasys.com) of San Jose, CA has acquired **Technology Resource Group Inc. (TRGI; www.beasys.com/trgi)**, a Java information technology education, mentoring, and consulting organization based in Maynard, MA.

Versant ODBMS Supports WebSphere

Versant Corp. (www.versant.com), a Menlo Park, CA object database management system (IDBMS) vendor, has come out with the Versant Enterprise Container (VEC) for WebSphere.

VEC is an EJB-compliant container that plugs directly into IBM's WebSphere application server, allowing the Versant object database to provide persistence for JavaBeans running on an IBM Websphere application server.

Applicant, HP Ink License Deal

Startup **Appliant Inc. (www.appliant.com)**, a Seattle-based developer of enterprise application performance management software, has signed a licensing agreement with **Hewlett-Packard Co. (HP; www.hp.com)** that allows HP to use Appliant's Java verifier technology for verification and testing of Java code used within HP products. Appliant has developed several unique technologies for monitoring and managing applications, including a Java verifier that keeps Java code free of bugs and viruses.

INSTINET
A REUTERS Company

Trading In Real Time With Java

Instinet to go live with one of the world's largest real-time Java-based trading systems

Philip J. Gill

INSTINET CORP., BASED in New York and founded in 1969, is perhaps best known as the leading provider of "after-hours" stock market trading. But the company, a wholly owned subsidiary of London-based Reuters PLC, is actually a leading provider of agency brokerage services worldwide, both during and after regular market hours, for 17 securities exchanges around the world.

This fall, Instinet is slated to go live with one of the largest production Java-based applications, a new Fixed Income Trading System. With a global reach, it will serve thousands of simultaneous customers with real-time brokerage services in the buying and selling of fixed income instruments.

As an agency brokerage, Instinet does not participate in any transaction; it remains neutral, providing brokerage services for the benefit of its customers—both buyers and sellers, institutions and individual investors. Trades are done anonymously through Instinet's global network infrastructure, which connects thousands of traders and portfolio managers to securities exchanges around the world.

In 1997, Instinet decided it wanted to branch out from its core business in stock trading to the global fixed income markets. These markets typically involve the buying and selling of government bonds and similar financial instruments.

The company formed a new division, the Fixed Income Group, and set about determining what kind of information technology (IT) would be required to meet the special needs of the fixed income markets.

Business Challenge

To begin with, the proposed system would have to provide three services: bidding, offering, and trading.

It would also have to take into account special business and technological requirements, including:

- *"Future proof" architecture.* Meaning, the system would need to implement an overall IT architecture that would be flexible enough to adapt over time. For instance, it would have to be easy to add or delete new functions and to support new users.
- *One-year time-to-market.* Instinet's early research determined that fixed income markets represented a significant opportunity for expansion of its electronic brokerage services, and that it needed to have a system up and running in one year to take advantage of that opportunity.
- *A level playing field for traders.* All traders had to have the same access to the same information simultaneously at all Instinet locations around the world.
- *Multi-platform.* Traders use a variety of desktop platforms, including UNIX workstations, NT systems, and PCs.

Architectural Solution

Instinet's overall solution was based on three primary architectural considerations.

First, Instinet realized it needed a three-tier or n-tier architecture consisting of a thin client on the desktop, a middle tier of application servers that both physically and logically shared business logic and locations around the world, and a back-end database server. This architecture, Instinet determined, would not only provide system responsiveness but also simplify management, security, and the ability to update.

Second, because Instinet maintains seven sites all over the world, communications would be largely asynchronous, to retain an adequate control on costs and allow for the most efficient use of the network resources.

Finally, any solution needed to be component-based and driven by an object model.

The actual development of the new Fixed Income Trading System began in May 1998. It is currently in beta testing and should go into production this fall, making it the largest real-time trading system to utilize Java.

According to Rachel Moseley, the Fixed Income Group's chief information officer (CIO), the choice of using the Java language and a number of Java-related technologies in building the new Fixed Income Trading System was clear from the start. "I felt that the platform independence was extremely important for our system, since traders have UNIX workstations and PCs, and we wanted them all to have as much flexibility as possible," explains Moseley. "The only thing that would do that was Java."

Implementation

Beginning at the top, or back-end database tier, Instinet chose Oracle Corp.'s Oracle8 relational database. Specifically, it is using Oracle8 Parallel Server to provide a consistent centralized database of all bid, offer, and trading information. All the back-end logic is written in C++.

In the middle tier, Instinet is using multiple instances of Persistence Software Corp.'s PowerTier application server. All the business and application

What Good Is An Object Tool Without An Object Database?

Attracting and retaining customers requires applications that are not only entertaining and informative, but also easy to use.

Today's business information is made up of not only text and numbers, but also images, motion, and sound.

Jasmine's object technology is designed for building enticing systems that you get to market faster, develop more rapidly, and gain productivity through reuse and integration of code, data and applications.

Only a pure object database like Jasmine inherently understands how to store and process complex data. Complex queries, such as fare comparisons for specific destinations and dates, are easily implemented with Jasmine.

Jasmine Works With Your Favorite Tool

- Sun:
  - Java Studio
  - Java WorkShop
- Microsoft:
  - Visual InterDev
  - Visual C++
  - FrontPage
  - Visual J++
  - VisualBasic
  - Visual Studio
- Symantec Visual Café
- Inprise JBuilder
- Bluestone Software
- Centura
- SilverStream
- SuperNova
- Dataviews Corporation
- Trellix

Just think how much more powerful your object tool would be with an object database like Jasmine™. With Jasmine's pure, object-oriented architecture, objects defined in your application tool (C++, Java, and others) are the same as the objects managed by Jasmine.

INFORMATION IN HARMONY™

You can use native language bindings, rather than learning and coding new non-object-based data access protocols such as JDBC. And Jasmine supports large amounts of complex data so you can deliver true, next-generation applications.

With Jasmine, objects never have to be converted to relational formats, so they run much faster. And there's no need to choose between object models—Jasmine supports both COM and Java. It even lets you use different tools at the same time. Jasmine's object infrastructure also integrates external data and applications, making them accessible from any application development tool.

That includes the object tool you're using right now. So call **1-888-7JASMINE** or visit **www.cai.com/ads/jasmine/dev** for your FREE, full-function Jasmine Developer Edition CD. And turn your object tool into an instant advantage.

COMPUTER ASSOCIATES®
Software superior by design.

Jasmine™

©1998 Computer Associates International, Inc., Islandia, NY 11788-7000
All product names referenced herein are trademarks of their respective companies.

logic at this tier is written in Java using the Java2 Enterprise Edition (J2EE) technologies, such as Enterprise JavaBeans (EJB).

On the desktop, traders have their choice of systems, which are loaded with a Java application that is installed, managed, and updated from Instinet's application and core servers.

The Java business logic was written using JBuilder, a rapid application development (RAD) tool from Inprise Corp., formerly Borland, of Scotts Valley, CA.

Hardware for both the core Oracle system and application servers are Ultra Enterprise servers from Sun Microsystems Inc., Mountain View, CA, running Solaris, Sun's UNIX operating system.

Communications between the different tiers occur over Instinet's IP-based Extranet using permanent virtual circuits and gateways into the Reuters External Services Network, a secure IP-based backbone.

Middleware plays a key part in making the system work. Instinet is using the TIB/Rendezvous publish/subscribe messaging software from another Reuters subsidiary, Tibco, to provide internal delivery of data between the back-end database servers and the distributed application servers.

The Java Messaging Service (JMS), in the form of Push Technologies Inc.'s Java-based SpiritWAVE messaging software, is used to deliver data from the mid-tier application servers to the traders' desktops.

Another Java-based messaging product, Castanet, from Marimba Inc. of Palo Alto, CA, is used to manage the distributed infrastructure and distribute software to traders' desktops and updates.

Mixing Java and C++

When the project was launched about a year and a half ago, Java wasn't quite ready to handle all the tasks Instinet envisioned its new Fixed Income Trading System would require. As Moseley explains, at that time, Java had a very bad image on Wall Street, primarily because of persistent rumors about performance problems.

As part of its risk-mitigation efforts, Instinet opted to combine Java with C++ to ensure adequate performance. "We limited our risk," said Moseley. "Depending on which version of Java you used, performance was still an issue a year ago."

"Right now, we're experiencing no performance problems with the system," she said. "In fact, we're very pleased with [it]."

Sun, she said, has been working diligently to improve Java performance, and she sees a very bright future for Java on Wall Street and other places.

And, given the continuing improvements Java is making, Moseley said the future for Java looks good. "A year or a year and a half from now, I would probably make a different decision [than combining Java and C++]," she adds.

Enterprise JavaBeans

A key part of the J2EE platform is the EJB component framework. Moseley said the use of EJB has improved the efficiency of the development team at Instinet, enabling this complex trading system to go from the drawing boards to production in about 19 months.

Application Server

Persistence Software's PowerTier application server was chosen primarily because of its EJB implementation as well, said Moseley. "We liked their EJB implementation, and, at the time, there really wasn't any competition."

Moseley said the fact that it supported both C++ and Java, the core languages for the new system, was another critical factor in its decision to go with Persistence's product.

The PowerTier application server also provides caching and synchronization capabilities. Its shared transactional caching capabilities help provide the real-time playing field the system requires, supplying all users concomitant access to the same bid, offer, and trading data.

Common Object Model

The PowerTier server was also instrumental in using a common object model for building the application, a key goal. Because it supports both C++ and Java, the Instinet development team was able to generate the core system logic in C++, while using the PowerTier/EJB version of the server to generate the business logic that resides in the middle tier, both from the same object model.

Oracle Database

The Fixed Income Group opted to use Oracle8 Parallel Server, because together, the Oracle database and Sun UNIX servers offer the best combination for high-performance applications, said Moseley.

"Surprisingly, Oracle is not the standard at Instinet," said Moseley. "The fact is that it's a very solid product, from a very solid company, and it has the performance we needed. Oracle with Sun hardware showed the best benchmarking results."

Using Oracle Parallel Server allows Instinet to operate the database as if it were one single, centralized server, but to have it physically distributed around its network of worldwide sites.

Java Migration

At its production launch, the new Fixed Income Trading System will use a combination of C++ and Java, although Moseley indicates that, over time, the system will migrate more and more toward Java.

"[When] we decided to go with Java, there was nothing like a real-time trading system in Java," said Moseley. "A lot of companies were still experimenting. I felt that Java had come a long way and that a lot of companies based their doubts more on a history that was maybe two or three years old. Java *has* come a long way since then." ■

Philip J. Gill is Associate Editor of *Java Report*.

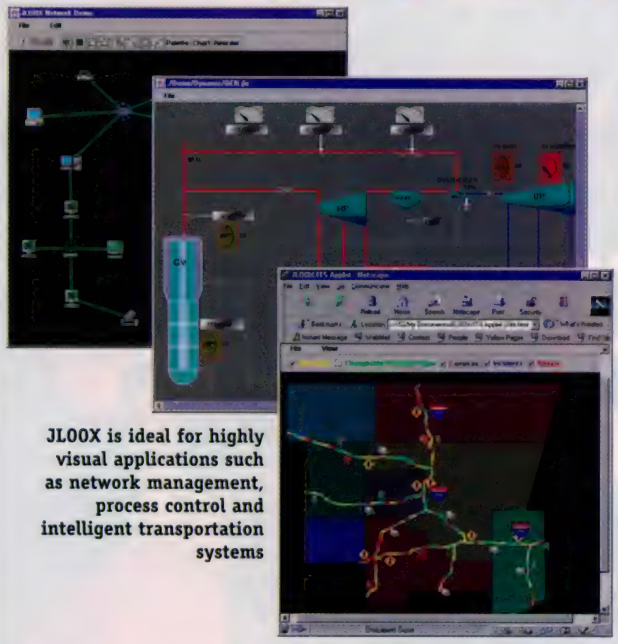
URLs

Instinet Corp.
www.instinet.com
Oracle Corp.
www.oracle.com
Persistence Software Corp.
www.persistence.com
Inprise Corp.
www.inprise.com
Sun Microsystems Inc.
www.sun.com
Tibco Software Inc.
www.tibco.com
Push Technologies Inc.
www.pushtechnologies.com
Marimba Inc.
www.marimba.com

Accelerate your Java™ Graphics Development Today

JLOOX

JLOOX is the best-in-class graphics toolkit for Java 2 developers. Based on Java2D, JLOOX provides a high-level 100% Java API that offers breakthrough performance and productivity gains for developers of graphics intensive Swing applications. It is ideal for applications requiring highly graphical displays such as networking and telecommunications,intelligent transportation systems, process control and simulation.



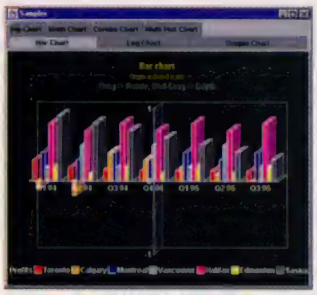
JLOOX is ideal for highly visual applications such as network management, process control and intelligent transportation systems

Using the JLOOXMaker interactive editor, complex graphics screens can be created quickly and easily.

Unique advantages:

- Includes an interactive editor for rapid creation and animation of graphic screens
- High-level intuitive API decreases development time
- Offers the rendering power of Java2D
- Pre-built collections of objects provide powerful data-visualization primitives
- Includes the award winning JClass Chart from KL Group
- Can be used with any IDE supporting Java 2
- Applications created with JLOOX have no runtime, royalty or deployment fees

Dynos are powerful data visualization, instrumentation and control objects.



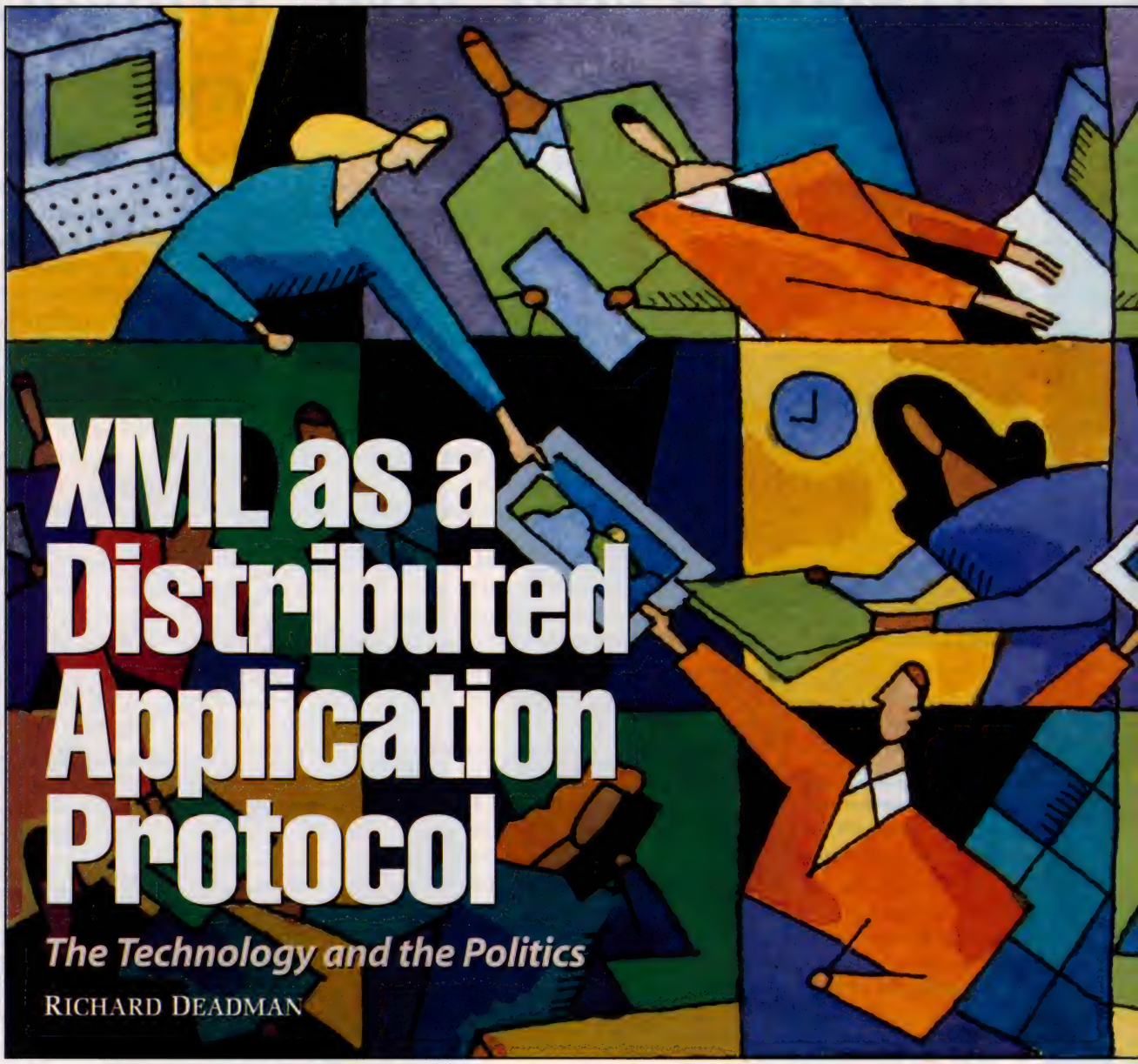
JClass Chart adds powerful graphing capabilities to Java

Accelerating Graphics Development™

LOOX Software Inc.

4962 El Camino Real, Ste. 206, Los Altos, CA 94022 • T:650-903-0942 F:650-903-9824 • www.loox.com

Copyright (c) LOOX Software Inc. 1999. LOOX is a registered trademark and JLOOX is a trademark of LOOX Software Inc. All other trademarks are the property of their respective owners.



XML as a Distributed Application Protocol

The Technology and the Politics

RICHARD DEADMAN

IT'S THE LATEST buzzword in the Internet application arena; XML, the open-standards child of SGML that promises to provide platform- and language-neutral data encapsulation and separate application logic from application data. It's hot. It's powerful. Everyone loves it.

But isn't this exactly what CORBA already provides?

We will explore why XML is so hot, show how XML can be used as a distributed computing protocol, and look at its advantages, disadvantages, and appropriate uses. We will also speculate on some of the non-technical forces driving the XML phenomenon.

XML is being touted as the perfect partner for Java. Java supports the development of Web-aware, platform-neutral applications, and XML is a platform-neutral document description meta-language. Like a Web startup, both technologies have great promise, and maybe greater hype. Small wonder that most of the XML tools becoming available are written in Java.

Brief Background

XML, for those just starting to pay attention, is a descendent of structured document markup languages such as Tex, troff, and quite directly, SGML. It consists of two basic parts, an optional structure definition (Document Type Definition or DTD) and the documents that use this definition to organize their content. Think of HTML where you can add your own tags for a set of related documents to use. This allows the content and structure to be separated. A third facility, the XML Style layer or XSL, allows presentation styles to be applied to any document of a particular flavor.

What are the advantages? Well, let's take a particular example. Suppose that, at present, corporate time sheets are created using an HTML editor. This is great, in that the creator is free to pick the most appropriate tool on the most appropriate platform for him or her. Also, anyone can view the time sheets with ubiquitous technology. But how does the corporation collect the information? Humans infer the

Robin Jareaus / Lauging Stock

day/project information by their placement within tables, but this kind of inference is very hard for a computer application to make accurately. However powerful they may be at crunching numbers, computers are not really very smart.

XML offers the computer help. We can define XML tags that state that we are now in a day with a day number attribute, and have embedded tags in that day for each project and the number of hours spent on that project. We can now define multiple style sheets for the XML document that allow payroll and managers to view the same document with different views. The advantages are two-fold: explicit structure for data manipulation and decoupling of the presentation from the data to allow for different views.

Wonderful, you say. Better structured documents than HTML can give us.

Of course, despite many predictions, XML will not cause HTML to disappear. There are just too many instances where the data being captured for presentation does not have enough "structural value" to justify the added cost of developing the DTD and XSL. Think of your average home page on the world of gum collecting. There are many cases where the information value or need for management is not

> ***"Of course, despite many predictions, XML will not cause HTML to disappear"***

sufficient to justify the cost of creating a structure and presentation layer. For these cases, HTML, with its predefined generic structure and presentation, is the logical choice. Indeed, HTML may become one of a list of predefined XML instances as it is modified slightly to fit the XML syntax.

The Next Step

Now that we have a platform-neutral document markup language, why not use it as the glue between applications? And because we are in Web land here, why not use it to describe data that must be shared between distributed applications?

Microsoft has taken this approach in the definition of CDF, an XML-based channel definition format that it released in March 1997, when PointCast and channels were all the rage. CDF defines channels and software distribution meta-information such as logos, abstracts, and modification dates. The last modification date is, for instance, an attribute string of format "yyyymmmddhhmmss" that indicates in Greenwich Mean Time when the channel was last changed. It is repeated in several elements because XML does not support element inheritance.

Hold Your Horses!

But wait a minute. Let's not pretend that XML invented platform-neutral data serialization. Since the days of DCE, frameworks have been developed for applications to exchange messages and data. The two most common examples of this are CORBA and RMI. They both seamlessly offer ways of moving an object tree to an application on another machine—and in many ways much better than johnny-come-lately XML. Although RMI allows for self-describing object serialization and is the simplest to use, I will concentrate on CORBA, because RMI, while platform-neutral, requires Java at both ends of the pipe.

It would be a simple matter to define the CDF as a CORBA IDL that supports a reusable date type. In fact, DTD to IDL mapping is much simpler than IDL to DTD mapping, because DTDs only support strings.

Here's the briefest background information on CORBA you're ever likely to read. Basically, CORBA is a distributed object framework. It creates local proxy objects for remote objects that your code wishes to talk to. When you send the local proxy object a message, including parameters, the proxy marshals the data to the remote system, calls the same method on the remote object, and returns the result back to you. In-out parameters, remote exceptions, one-way

messages, contexts, security frameworks, naming, trader, transaction, and other services also exist, of course.

The magic of CORBA is that your client can be a Java applet running on a Linux PC, sending messages to objects in a Smalltalk application on a Solaris server. And the parameters can be basic types (int, String, float, etc.) or any object whose class has an IDL structure mapping. What is IDL? Just as XML allows for a DTD to share the structure of the tag sets, the Interface Definition Language is the language-neutral shared definition of the object types (and remote messages they support). It is what it says it is, a language-neutral interface definition language. Tools are supplied by CORBA vendors (and JavaSoft) that create stubs and skeletons for IDL specifications.

> *"XML is cool and, as Marshall McLuhan noted, cool is hot"*

CORBA is not perfect for all applications. Although free ORBs are available, to get the services or support required often involves the licensing of both developer and deployed application, with all the license key management overhead and costs that involves. And unlike Java's RMI, automatic behavior marshaling is not supported. That is, if the IDL says an object of type User is being sent across the wire, and you send an instance of Student (subclass of User) into the remote method, the object will morph into a User at the remote end, probably with incorrect behavior for some methods. Not that XML handles this any better. What XML does handle better, however, is:

- *Large volumes of data.* Most XML parsers support the SAX interface that allows the document to be streamed in and processed without a full object model being built in memory. Obviously, there are times when this is absolutely necessary.
- *Persistent management of text trees.* An XML document is simply a stream of bytes, and so can be stored in a file for later retrieval using the correct version of the DTD. CORBA only exposes object trees, and while these can also be serialized to data files, they are brittle in that changes to class definitions can cause deserialization errors. Java does not allow for class versioning in the way that DTDs may be versioned.
- *Manual intervention.* Let's not forget the different routes that have converged here. XML is a child of the human-readable text markup evolution. CORBA emerged from DCE as a way to extend computer processing between machines. If all else fails, an XML stream can be visually inspected and parsed by a human.

XML can be thought of as the text processor's answer to data management. It is designed to encode the human-oriented documents that still drive many business processes. CORBA was designed by programmers to handle distributed system interactions and, as such, suffers from the complexities that go along with lower-level programming issues. In a way, XML can be seen as a revenge on programmers for developing a data distribution system that was too abstract and difficult to program.

Feature Checklist

Table 1 is a checklist of capabilities of both XML and CORBA. I've also included RMI for completeness. By determining your needs, you can use this checklist to help determine the most appropriate technology for distributing data among your application. This is not intended to be a complete checklist of all distributed system needs or capabilities. Caveat emptor.

Politics

So, if XML has certain benefits but also certain drawbacks for building distributed systems, why is there so much pressure currently to apply it to all problem domains? Part of this phenomenon stems from our natural inclination to try to see how far we can push any new "cool" technology. But this is only part of the explanation.

Take Microsoft, for example. It is always dangerous to speculate too much in the political arena. Note, however, that Microsoft has made XML one of its linchpins in its drive toward Web-based open systems support. Of course there are rumblings that Microsoft is trying to put its own proprietary spin on the technology, but let's give Bill and company the benefit of the doubt.

CORBA has never been endorsed by Microsoft. Microsoft was an early member of the OMG long before the Web was hot. But they always sat on the fence about supporting CORBA and finally decided that CORBA competed too directly with their DCOM (now called simply COM) object model plans. DCOM was the distributed extension of COM, which itself was an evolution of OLE, a technology for allowing windows applications to interact with each other. Some say that Microsoft wanted to control the desktop environment and saw DCOM as a way to extend its desktop presence to a whole network, thereby mandating its operating systems as the required de facto standard. Microsoft couldn't support CORBA too directly without sabotaging their DCOM message.

Assuming that this analysis is true, what happened next? Well, the Web came out of nowhere and threw Microsoft's plans for Blackbird, its proprietary Information Superhighway technology, into disarray. For several months, Microsoft appeared to be reacting like IBM at the beginning of the PC revolution. Finally, they ate some crow, or maybe blackbird, and embraced the Web.

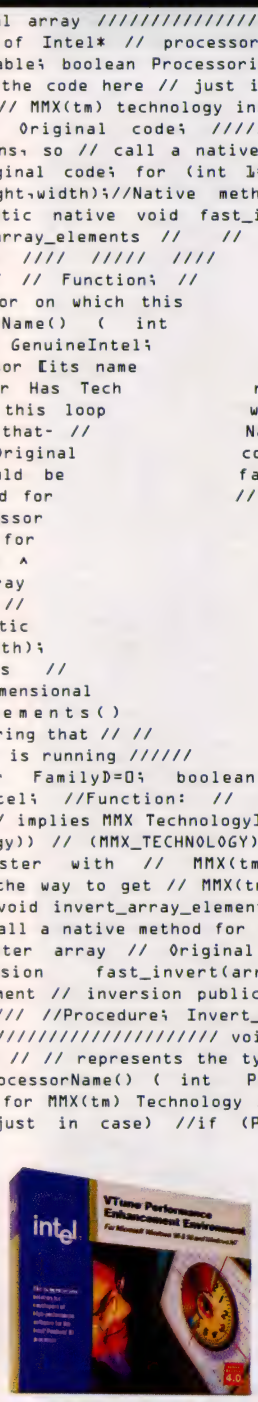
Some say the Web is antithetical to Microsoft's philosophy of controlling the technology behind its products. In the Web world, open is good, vendor-neutral is good, OS- and language-independence is good. So, while ActiveX morphed and limped out of OCX's, Java became ubiquitous. Open source and Gnu public licenses became the flavor du jour. Finally, Microsoft needed to get in the open game when it came to Web-based data standards. The W3C was not going to accept submissions for a channel definition format based on DCOM. But publishing it as a CORBA IDL would be treasonous to its DCOM technology.

Enter XML. It's an open, vendor-neutral, OS-independent standard spearheaded by the W3C. And it doesn't give the image of betraying DCOM. In fact, paradoxically, it is precisely XML's inability to specify a remote messaging interface (the core of both CORBA and DCOM) that make it an acceptable remote data format to Microsoft.

<Optimize your code>

and make it perform more powerfully // on the latest Intel® processors

[check those apps! <!!~]



//Here's the easy way to increase your code's power, especially if you're pressed for time. It's the VTune™ Performance Enhancement Environment 4.0. All the tools you need to take advantage of the power of the latest Intel processors, including the Pentium® III processor. The VTune Performance Analyzer quickly identifies performance bottlenecks and delivers priceless tuning advice. The high-performance Intel C/C ++ Compiler features support for the Pentium III processor's Streaming SIMD Extensions, and plugs into Microsoft Visual Studio.* And if that wasn't enough to whip your code into perfect shape, there's the Intel Performance Library Suite and the Intel Performance Training Center. Not surprisingly, this simple way to add extra muscle to your apps comes from the people who know the processor best. So optimize your code and make it as powerful as you imagined. < Close dialogue now >

Download a free, 14-day evaluation @ http://developer.intel.com/vtune

intel®

©1999 Intel Corporation. All rights reserved. Intel and Pentium are registered trademarks, and VTune is a trademark of Intel Corporation.
*All other brands and trade names are the property of their respective owners.

Table 1. **Capabilities of XML and CORBA.**

CAPABILITY	DESCRIPTION	XML	CORBA	RMI	WINNER (EXCLUDING RMI)
Platform language independent	Can the client and server be on almost any OS, hardware, and application language?	Yes	Yes	*No (only supports Java)*	Tie
Handle human readable text	Can the document be viewed and edited with low-tech tools?	Yes	No	*No*	XML
Handle non-string data types	Is there native for integers, floating point numbers, booleans, etc.?	No (each DTD may define its own mapping to string-based representation, but there is no validation or parsing support or reuse. **Note:** Currently, work is underway to add this to XML through the Document Content Definition proposal)	string, short, int, float, double, boolean, byte, char, enums, structs (limited 0-0 support), union	*all serializable objects*	CORBA
Huge data set management	Can the data be larger than the allowable application memory?	Yes	No (although with CORBA, data is usually only retrieved on an as-needed basis as part of a remote conversation)	*No (see CORBA note)*	XML
Schema versioning	Can an application handle two data streams that were created using different versions of the schema definition?	Yes	No	*No*	XML
Distribution support	• transparency • remote messaging • lookup facilities • security • transactions, etc.	none (simple messaging can be achieved by parsing transmitted XML documents)	full	*partial (naming, remote method invocation, mobile code security) EJB, JTS, JNDI, etc., offer most CORBA services*	CORBA
Object-oriented	Can the data structure have behavior associated with it? Is inheritance and reuse supported?	No	partial OO mapping; IDL to Java generator creates mobile classes without behavior. Only remote objects can inherit	*Yes*	CORBA
References	• If node A refers to node B, can node B refer to node A? • Can nodes B and C both refer to node A? • Lazy pointers	No cyclic support; Support for shared and lazy references	No	*support for cyclic and shared references; no support for lazy pointers*	XML
Integration with application object model	Does the technology allow the application object model to be used as is?	No. XML parsers generally support DOM- or SAX-based object models. Use of application object model requires mapping functions or adapter layers	somewhat (the IDL to Java generator creates structure classes that must then be used)	*full (any application object which is serializ-able can be passed as a remote parameter or returned as a return value)*	CORBA

Or so the theory goes. I have no knowledge of Microsoft's decision-making process and so all of this is, obviously, speculation. Flame shields up.

More Politics

This is not the whole picture. Sun seems to add support for every new buzz word to Java as soon as it appears. It has taken a couple of years for Sun to resolve its apparent splintering from the CORBA camp with the release of RMI. Now they are pushing a Java XML standard extension and talking of using XML to make Enterprise JavaBeans (EJBs) more portable. Of course, as we saw earlier, there are valid domains for XML, and XML and Java have hit it off from the beginning. For class-version robust specification of persistent data, such as EJBs configuration, XML is a good choice. So Sun's work may not be as contradictory as it may first appear.

IBM, as well, has come out with a lot of XML support and components. This appears, however, to be more of a support for OS-neutral standards than a push to extend XML into the distributed messaging market. With so many of its own operating systems to support, as well as a stodgy image to update, IBM has a lot at stake in supporting both Java and XML.

Even the OMG has had to support XML, with some obvious reservations. But with the OMG, that is probably more of an effort to appear open to new technologies and directions than any belief that it may be a good replacement for the core CORBA technology. Better to say "XML has its uses, and here's how it can fit with CORBA," than to put your hands over your ears and chant "I can't hear you."

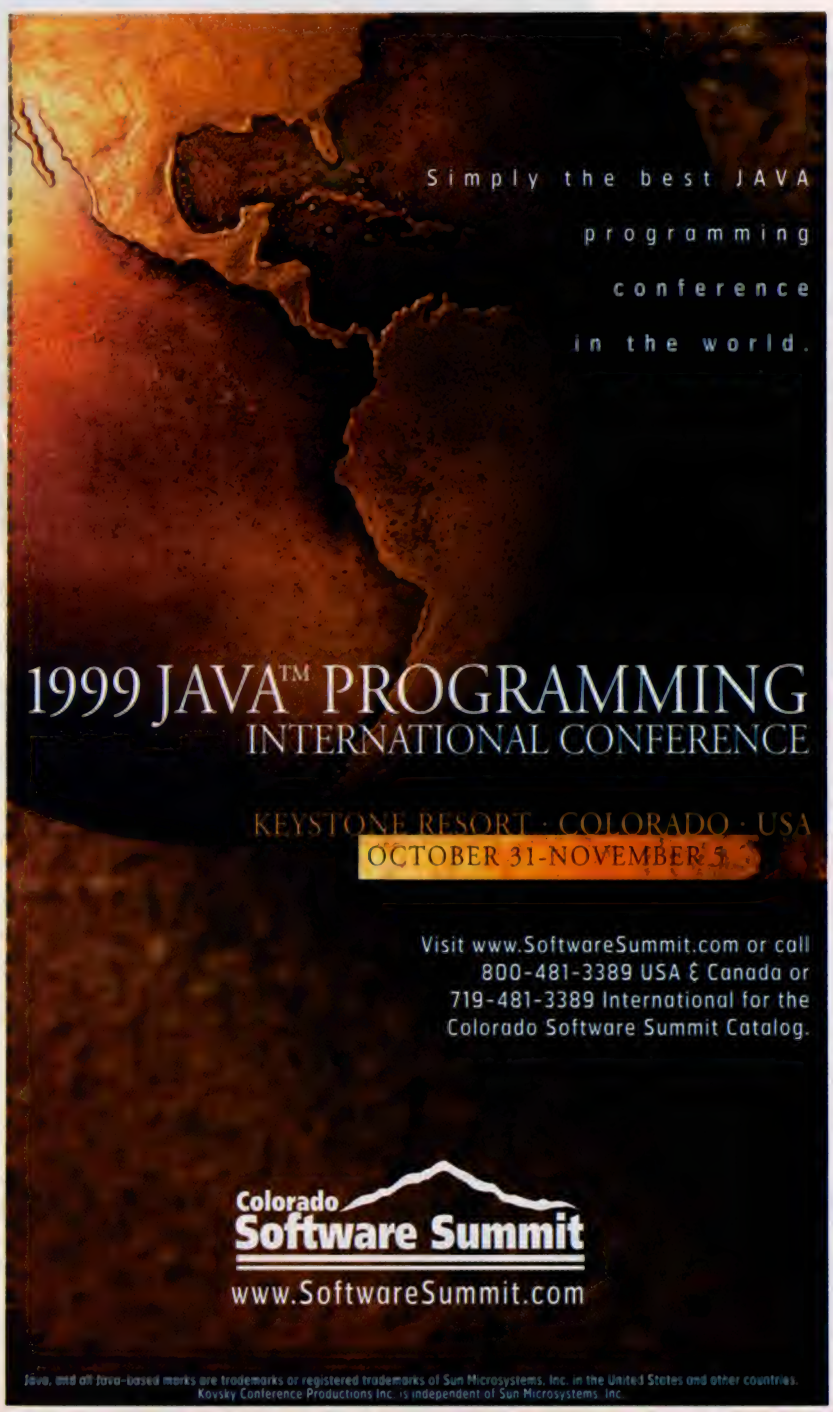
Wrap Up

XML is cool and, as Marshall McLuhan noted, cool is hot. It offers a simplified, Web-friendly data markup language that provides domain-specific tagging and promises to make the maze of documents on the Internet easier to manage, search, and navigate. As a distributed data format, it has its advantages but also many disadvantages when compared to the other Web-centric distributed data standard: CORBA. It handles huge data sets of human-readable text data with lazily-followed pointers exceptionally well. It does not provide a distributed messaging framework, nor support machine-readable non-text data. The reason for its push into the distributed computing arena comes from both a desire among geeks to push new technologies as far as they will go, as well as political desire by some organizations to support open standards without threatening their own technologies.

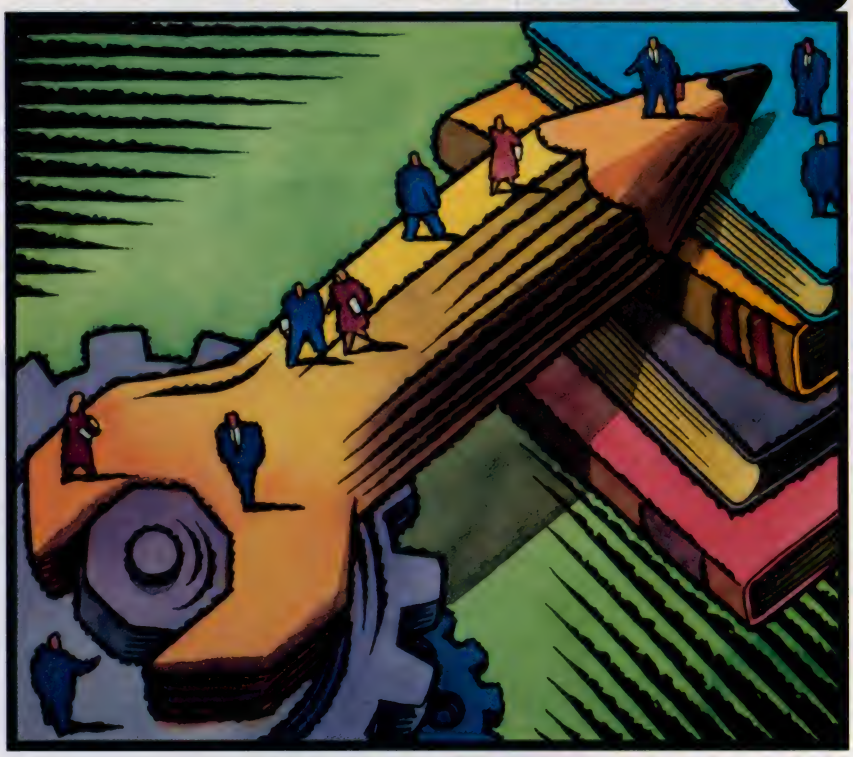
There are many application arenas where XML provides huge advantages, especially in document management and stream-based data manipulation. Other applications may not benefit as much from XML as they would from other open Web technologies better suited to their needs. Hopefully, I've helped architects and designers get a better understanding of when it is and is not appropriate to use XML as part of the system architecture. ■

Richard Deadman is a contractor in distributed Java systems in Ottawa, Canada. Currently he is serving as Chief Architect, Distributed Systems at Xia Information Architects. He can be contacted at rdeadman@magma.ca.

Quantity reprints of this article can be purchased by phone: 717-399-1900 ext. 139 or by email: sales@rmsreprints.com.


Simply the best JAVA
programming
conference
in the world.
1999 JAVA™ PROGRAMMING
INTERNATIONAL CONFERENCE
KEYSTONE RESORT · COLORADO · USA
OCTOBER 31-NOVEMBER 5
Visit www.SoftwareSummit.com or call
800-481-3389 USA & Canada or
719-481-3389 International for the
Colorado Software Summit Catalog.
Colorado
Software Summit
www.SoftwareSummit.com
Java, and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.
Kovsky Conference Productions Inc. is independent of Sun Microsystems, Inc.

Error Handling with XML



David Chen / SIS

Preserving and sharing detailed information among components can greatly assist in application debugging and system recovery

STEVEN JONES

ERROR HANDLING IS important for any serious application and can be especially tricky for distributed applications. RPC Middleware platforms such as CORBA, RMI, and EJB provide exceptions for reporting errors across process boundaries, and this can go a long way toward a comprehensive error handling approach. However, these methods may not suffice for passing detailed, contextual error information that can greatly assist in diagnosing problems. In distributed systems that do not directly support the use of exceptions (e.g., asynchronous messages, low-level sockets, or http), or when application components are more loosely coupled and communicate through other means (e.g., files, databases), a more structured and well-defined mechanism is needed to preserve valuable debugging information when errors occur. Finally, sharing extensive error information between languages without the use of COM or CORBA is quite difficult and often reduces to minimal implementations (e.g., passing low-level error codes).

The following presents a unique solution for handling errors in a language- and platform-independent way using XML. A document type definition, or DTD, is provided that describes how to capture important error information so that it can be shared more readily within these scenarios. Several XML document examples are also included that conform to this DTD and that demonstrate the value of this approach. Once captured in XML, the error information can be utilized by a variety of XML-friendly runtime environments, including browsers, Perl scripts, and Java components. To best illustrate XML's use in the Java environment, several Java classes are provided that help convert Java's own Exception objects to and from the language-independent XML representations. Using XML to capture and share important error information can enhance your overall error-handling approach. XML can be used to increase the

Find hidden problems automatically with NuMega™ DevPartner™ for Java!

Serious problems can lurk quietly in your Java™ code, and if left undetected, can cause performance and reliability issues after your software is deployed. **NuMega™ DevPartner™ for Java** from Compuware® automatically finds performance bottlenecks and hidden runtime errors in your Java code, which can save you days of valuable development time.

DevPartner for Java is a suite of developer productivity tools that accelerate software development and improve the performance and reliability of your Java code. DevPartner for Java includes **TrueTime** — the award-winning performance analyzer that automatically pinpoints performance bottlenecks, with or without source code; **TrueCoverage** — the code coverage analyzer that automatically measures and tracks tested code, untested code and code base stability; and **JCheck** — the visual thread and event analyzer that automatically detects thread errors, runtime errors, leaks, unhandled exceptions and other potentially serious problems. Plus, all DevPartner for Java tools have powerful multi-VM switching capability, which can help you find behavioral problems that occur when your code runs in different virtual machines.

Find hidden problems in your Java code with NuMega DevPartner for Java, and deliver the fastest, most reliable Java applications and components with confidence!

Supports Java 2

Accelerating Software Development
with
NuMega DevPartner for Java from Compuware

NuMega DevPartner for Java includes:
NuMega TrueTime™ Java Edition — Automatic Performance Analysis and Optimization
NuMega TrueCoverage™ Java Edition — Automatic Code Coverage Analysis
NuMega JCheck™— Visual Thread and Event Analysis

Call today to request your copy of the technical white paper
"Effective Unit Testing for Java Developers"
or download the paper now by visiting
www.compuware.com/numega

1-800-4-NUMEGA
www.compuware.com/numega
30-Day Money Back Guarantee

JDJ Java Developer's Journal World Class Award

COMPUWARE®
People and software for business applications™

Copyright © 1999 Compuware Corporation. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.
Java and all Java-based marks are trademarks of Sun Microsystems, Inc. in the United States and other countries.

flexibility you have in processing errors and to enrich existing error processing mechanisms, thereby improving the robustness of your entire application or system.

Error Handling in Distributed Architectures

Often overlooked in the grand design of architecture, error handling is typically "achieved" by logging messages to a flat text file. This approach is actually encouraged by the great number of systems management tools that are designed to parse and monitor log files, raising alarms when certain predefined tags are encountered. Logging to a flat file is basically a simple and passive technique for handling errors. It can be refined, of course, by using abbreviated error codes and message lookup tables, by applying message filters, and by exploiting error severity; however, even with such enhancements, simple message logging is not sufficient for client/server or n-tiered applications.

The effect of resorting to simple message logging or passing error codes between processes is a reduction in one's ability to accurately diagnose faults. Errors logged in one place may not be visible to users in another. For example, the Web user may never know the reason that the CORBA object failed because it was not preserved by the Java servlet that actually made the call from within the Web server. The log file generated by components in the middle tier may not be easily parsable by humans and may require special scripts or programs to search for and investigate fault history or recurring patterns. After all, the goal of an error-handling approach is not to devise a clever error strategy but to ensure the fastest and most appropriate response to a software failure.

Distributed applications are inherently more complex than stand-alone programs, sometimes employing many components written in different languages and communicating over one or more protocols (e.g., http, IIOP, COM). Distributed applications must employ a straightforward and consistent mechanism to pass error information between the various components and/or processes. I'll introduce such a mechanism and explain the benefits of a standardized approach for error handling in distributed applications.

Difficulties Facing Developers

In crafting an error-handling strategy, most developers turn first to features of the target implementation language; however, beyond exceptions, which are supported in popular object-oriented languages like C++ and Java, there is a serious lack of support for robust error handling. For example, the developer must devise a custom approach for capturing and preserving any appropriate contextual information when handling an error. Contextual information can include anything that may help in understanding and properly reacting to the actual error, from a simple timestamp to the current method or function name; from the local state, such as a stack trace, to any system or global state. How can this information be saved and associated with the error as it is propagated up through the call stack or from process to process?

Even if this information could be captured, the lack of any standard error format makes it difficult to share errors among components on different platforms or written in different languages. These languages and their respective development environments do not provide sophisticated built-in tools that assist either the developer or the runtime user in handling errors.

Along Comes XML

The rise in popularity, acceptance, and support of XML has opened up new opportunities for software developers. Architects are continually finding new uses for XML. It certainly can play a key role in the design of a robust error-handling strategy.

LISTING 1.

Error.dtd.

```
<!—
*************************************************************
 File: Error.dtd
 Description:This file contains the grammar for XML Error elements.
*************************************************************
—>

<!— XML Errors contain optional context text, a type, message, or
message number, and an optional stack trace —>
<!ELEMENT Error (SystemContext?, ApplicationContext?,
      Type, (Message | MessageNum),
      StackTrace?)>
<!— date/time stamp —>
<!ATTLIST Error ReportedAt    CDATA     #IMPLIED>

<!—Any runtime information regarding the system and its environment that can provide help in the investigation of the error —>
<!ELEMENT SystemContext (#PCDATA)>

<!— Any runtime information regarding the module, method, or
function that can provide to help in the investigation of the error —
>
<!ELEMENT ApplicationContext (#PCDATA)>

<!— Type has name attribute and a set of one or more supertypes —
>
<!ELEMENT Type (Supertypes?)>
<!ATTLIST Type Name       CDATA     #REQUIRED>

<!— Recursive declaration: Type can contain one or more supertypes
—>
<!ELEMENT Supertypes (Type)+>

<!— Plain data —>
<!ELEMENT Message (#PCDATA)>

<!— Lookup value —>
<!ELEMENT MessageNum (#PCDATA)>

<!— StackTrace is composed of one or more StackFrames —>
<!ELEMENT StackTrace (StackFrame)+>

<!— Represents a single frame on the stack —>
<!ELEMENT StackFrame (#PCDATA)>

<!— EOF —>
```

Java™ 2 Platform Enterprise Edition

Java™ 2 Platform Enterprise Edition (J2EE) is the single most important standard to emerge since Web applications became the world's new way of doing business. Understanding this standard and how to implement it is critical to the success of your Web applications. Now you can learn what you need to know about the J2EE platform – and how to make it all work for you.

Attend a FREE half-day educational seminar – *Java™ 2 Platform Enterprise Edition for the Real World,* brought to you by SilverStream Software and Sun Microsystems. Through technical demonstrations and instruction from industry experts, you'll gain a better understanding of the impact of the J2EE platform, the details of the specification and how you can use it to quickly deliver robust Web solutions.

WHO SHOULD ATTEND?

Whether you're a developer or architect who's already knowledgeable about the J2EE platform, or you're just starting to investigate it, this FREE half-day educational seminar is right for you.

WHY ATTEND?

- Learn what you need to know about using the J2EE platform to implement real-world applications
- Discover how and when to use J2EE technologies such as Enterprise JavaBeans™ components (EJB), Servlets, JavaServer Pages™ components (JSP), Java™ Transaction Service (JTS) and Java™ Foundation Classes (including the GUI toolkit, JFC/Swing)
- Receive a Beta of the newest version of the SilverStream Application Server
- Get valuable materials to reference as you implement your applications
- Learn how to access additional resources such as training and services to help you get started

REGISTER TODAY FOR A FREE HALF-DAY EDUCATIONAL SEMINAR

www.silverstream.com

Or phone 1.888.823.9700 (toll-free in the U.S.) or 1.508.788.3260 (outside the U.S.)

LEARN HOW TO DELIVER REAL-WORLD APPLICATIONS WITH THE J2EE PLATFORM.

Copyright ©1999 SilverStream Software. Inc. SilverStream is a registered trademark and SilverStream Designer is a trademark of SilverStream Software, Inc. Sun, Sun Microsystems, the Sun logo, Java, Enterprise JavaBeans, and JavaServer Pages are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Sun microsystems | SilverStream®

erarchical structures. In fact, there are few places it seems where XML is not the appropriate technology for representing structured data, and error information is no exception. XML is inherently cross-platform and is accessible by many popular programming and scripting languages. It is universally supported by all the major players, including Sun, Oracle, Microsoft, and IBM. These and other vendors have already begun to supply tools for creating and parsing XML documents, and in most cases, the tools are freely available and relatively simple to use.

So what does this mean in terms of error handling for sophisticated applications? First, the platform-neutral aspect makes it ideal for representing errors in components deployed in a heterogeneous runtime environment. For example, legacy AS/400 applications can represent errors in the same manner as COM-based components running on the Web server and as script-enabled Web pages on the thin-client desktop! Second, the accessibility from many languages allows components to share error information more easily. The availability of language-specific parsers makes creating and interpreting XML-based errors easy. Finally, the very structure of XML itself, with self-describing tagged data, makes capturing and comprehending contextual information straightforward.

To create an XML Error solution, the tag structure or grammar of any individual error must be defined. This is analogous to defining a single class structure that will be used to create many individual object instances. The grammar is captured in a special document called the DTD. XML parsers can use the DTD to validate XML documents—this is a useful feature for ensuring the error information generated by any component is correct and complete. As suggested earlier, a component might want to capture contextual information on detecting an error and store that with the error message. In Listing 1, a DTD shows the grammar for a detailed XML Error.

Using this simplified DTD grammar, any application can create an XML Error. For example, a Java program can embellish the information available from any Java Exception object with available contextual information (see Listing 2).

But the beauty of XML is that there is nothing specific to any programming language. For example, a CGI Perl script could have created the following XML Error and returned it to the browser in response to a failed request. The error information can currently be understood by Microsoft's Internet Explorer, which handles XML natively, providing a convenient tree structure that allows the user to study the error right from the desktop. This result is infinitely more useful than a generic Web server error (see Listing 3)!

LISTING 2.

SimpleXMLError.xml.

```
<?xml version="1.0" encoding="UTF-8"?>

<!—
**********************************************************
 File: SimpleXMLError.xml

 Description:
  This file demonstrates how a single Java Exception or Error can be
  represented in XML. Here a java.io.FileNotFoundException has
  been represented in XML. The entire DTD grammar is used here in-
cluding optional fields.
**********************************************************
—>

<!DOCTYPE Error SYSTEM "Error.dtd">

<Error ReportedAt="Wed 02/04/1999 9:52:26 PM EST">
   <SystemContext>
      Program started at: Wed 02/04/1999 9:50:00 PM EST
      Duration: 00:02:26
      Thread info: [main:5:main]
      Program name: Test
      Free memory: 311432
      Total memory: 1048568
      Occurrences: 1
   </SystemContext>
   <ApplicationContext>
      Method: run()
      Parameters: i=10
      Debug: true
   </ApplicationContext>
   <Type Name="java.io.FileNotFoundException">
  <Supertypes>
     <Type Name="java.io.IOException">
     <Supertypes>
        <Type Name="java.lang.Exception">
         <Supertypes>
            <Type Name="java.lang.Throwable">
             <Supertypes>
                <Type Name="java.lang.Object">
                </Type>
             </Supertypes>
            </Type>
         </Supertypes>
        </Type>
     </Supertypes>
    </Type>
  </Supertypes>
   </Type>
   <MessageNum>
      <!— This number can be used to resolve an actual string mes-
sage —>
      12345
   </MessageNum>
   <StackTrace>
      <StackFrame>
         <![CDATA[
     at java.io.FileInputStream.open(Native Method)
     ]]>
      </StackFrame>
      <StackFrame>
         <![CDATA[
     at java.io.FileInputStream.<init>(FileInputStream.java:68)
     ]]>
      </StackFrame>
      <StackFrame>
         <![CDATA[
     at ErrorTest.main(ErrorTest.java:7)
     ]]>
      </StackFrame>
   </StackTrace>
</Error>

<!— EOF —>
```

YOUR CEO WANTS RELIABILITY, AVAILABILITY AND SECURITY.
YOUR CIO WANTS HIGH PERFORMANCE.
YOUR COO WANTS BROAD FLEXIBILITY.
YOUR CUSTOMERS WANT IT YESTERDAY.

HERE'S WHAT YOU WANT.

Our customers wanted all of the above, too. And they got it with Sybase Enterprise Application Server.™ Check out the success of Blue Cross/Blue Shield, First American, and others at www.sybase.com/easerversuccess. Or call Sybase at 1-800-8-SYBASE. And make everybody happy. Especially yourself.

SYBASE®
INFORMATION ANYWHERE.™

©1999 Sybase, Inc. All rights reserved. All trademarks are the property of their respective owners.

FORMU

ANNOUNCING SUPPORT FOR SWING, EXCEL 2000, CHARTING, AND MORE!

NEW 7.0 VERSION

- *NEW IN 7.0!* Read/write Excel 2000 files
- *NEW IN 7.0!* Support for 325 built-in functions
- *NEW IN 7.0!* Accepts dates through year 9999
- *NEW IN 7.0!* Workbooks expanded to 32,768 columns, 32,768 sheets, and more than a billion rows
- *NEW IN 7.0!* Java 2 support
- *NEW IN 7.0!* Swing support
- *NEW IN 7.0!* Java 2D support
- *NEW IN 7.0!* Model/View/Controller architecture
- *NEW IN 7.0!* Excel-compatible 2D charting
- *NEW IN 7.0!* Undo and redo
- *NEW IN 7.0!* InfoBus support
- *NEW IN 7.0!* Specialized, pluggable recalculation engines optimized for different situations
- *NEW IN 7.0!* Support for developer-specified, add-in worksheet functions
- *NEW IN 7.0!* Thread pooling support for more efficient performance
- *NEW IN 7.0!* Modular .jar files. Deploy only the functionality that is required

JAVA

USE SPREADSHEET

SCALABLE CALCULATION ENGINE • DATABASE REPORTING TOOL • DA

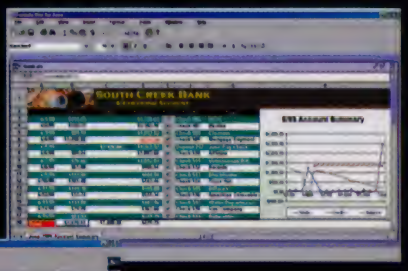
PRODUCT AND PRICING INFORMATION:
800 - 922 - 9665 • SALES@TIDESTONE.COM

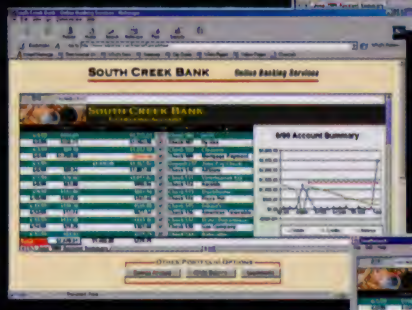
Tides
TECHNOL

Copyright © 1999 Tidestone Technologies, Inc. All rights reserved. Tidestone, the Tidestone logo, Formula One, and Internet Spreadsheet are trademarks of Tidestone Technologies, Inc. Java, 100% Pure Java, and all Jav

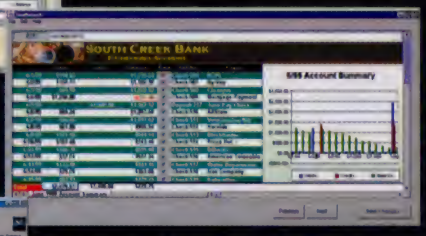
LA ONE™

Use Formula One as a standalone spreadsheet application on any desktop . . .





. . . or a JavaBean component for building servlets, JSP, applets, and Java applications . . .



. . . even place Formula One on a server to perform calculations, interact with databases, receive input from clients, and distribute spreadsheets or other tabular data as HTML.

The Internet Spreadsheet,™ server to client!

Formula One's workbook designer can be used as a full-featured standalone spreadsheet application by end users or developers to create and format sophisticated spreadsheets. It can even be launched from the Formula One applet or JavaBean with the press of a function key.

Speaking of JavaBeans, Formula One, now in 7.0, provides developers with a robust JavaBean component with more than 400 properties, methods, and events.

Formula One also offers the ultimate in versatility. It connects to a wide range of databases through built-in JDBC methods for robust database reports, provides a scalable calculation engine for fast computations, and delivers countless data formatting options, even several Excel compatible 2D charts. Formula One can also distribute data in a variety of formats, including its optimized .vts file format, Excel .xls files, or as HTML to thin-client computing environments.

Put Formula One to the test today! Visit www.tidestone.com/InternetSpreadsheet/ to download a free 30-day trial.

S ON THE INTERNET

PRESENTATION • DATA DISTRIBUTION • READ/WRITE EXCEL WITH JAVA

ne
TES

Demos and Full-Featured, Free Trials:
www.tidestone.com/InternetSpreadsheet

marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All other trademarks are property of their respective owners. Specifications subject to change without notice.

Returning now to distributed applications, let's explore a simplified but real-world example using Java and RMI. We will create a simple Java server program that, on catching an exception, creates and returns an XML Error to the calling client program. Java's Exception object provides many of the fields of the XML Error, so the creation of an XML Error in this example is trivial.

RMI is actually an interesting choice for this example because the underlying serialization technology supports sending objects back and forth over the wire (i.e., pass by value). Because Java's Exception objects can be passed between processes, the RMI-based distributed architecture can inherently support a more robust error-handling strategy. However, RMI currently only works with Java client and server programs, which leaves out many other popular and appropriate languages from the error-handling strategy (ignoring the RMI-over-IIOP work in progress). But even within Java programs, RMI does not handle its own errors—specifically its Exceptions—fully, as desired.

When a server program throws an Exception object to the client, as advertised in the published (Remote) interface, no stack trace information from the calling process is preserved. The client, on printing the stack trace of a server-generated exception, will find a local trace only. Thus, one of the most critical pieces of contextual information is made unavailable to the now disabled client and may perhaps be lost entirely if the server did not take

LISTING 3.

Example Error.xml.

```
<?xml version="1.0"?>
<?DOCTYPE Error SYSTEM "Error.dtd"?>
<Error ReportedAt="Wed 02/04/1999 11:22:46 PM EST">
  <SystemContext> Current time=Thu April 1, 1999 </SystemCon-
text>
  <Message>Server timeout: retry limit exceeded</Message>
</Error>
```

LISTING 4.

Client class in the default package.

```
1.  // SampleClient.java
2.  public class SampleClient {
3.     public static void main(String args[]) {
4.        try {
5.           SampleService service =(
             SampleService)java.rmi.Naming.lookup(
             "//localhost/SampleServer");
6.           service.doIt(1);   // throws XML-based exception
7.        }
8.        catch ( XMLException exception ) {
9.           exception.printStackTrace();   // output is specific
                                            //to client
10.
11.          // handle XML Error here
12.          if ( exception.isPreserved() == true ) {
13.             FauxException fauxException =
                 new FauxException(exception.xmlError);
14.             fauxException.printStackTrace();  // server side
                                                  //trace
15.          }
16.       }
17.       catch ( Exception ignore ) {  // ignore all others for this
                                        //example
18.       }
19.    }
20. }
```

LISTING 5.

Server classes in the default package.

```
21. // XMLException.java
22. public class XMLException extends Exception {
23.    public String   xmlError; // ignore encapsulation
                                 //for brevity
24.
25.    public XMLException(String message) {
26.       super(message);
27.    }
28.    public void preserve(Object appContext) {
29.       // uses utility class to create XML document
30.       xmlError = ErrorToXML.convertToString(this, appContext);
31.    }
32.    public boolean isPreserved() {  // convenience only
33.       return xmlError == null ? false : true;
34.    }
35. }
36.
37. // SampleService.java
38. public interface SampleService extends java.rmi.Remote {
39.    void doIt(
       int parameter) throws java.rmi.RemoteException,
       XMLException;
40. }
41.
42. // SampleServer.java
43. public class SampleServer extends
    java.rmi.server.UnicastRemoteObject
44.                   implements SampleService {
45.    public SampleServer() throws java.rmi.RemoteException { }
46.
47.    public void doIt(
       int parameter) throws java.rmi.RemoteException,
       XMLException {
48.       if ( parameter != 0 ) {
49.          XMLException xmlException =
             new XMLException("This is an example exception");
50.          xmlException.preserve(
             "in doIt(), parameter=" + parameter);
51.
52.          throw xmlException;          // create XML Error and
                                          // throw exception
53.       }
54.    }
55.    public static void main(String args[]) {
56.       try {
57.          ErrorToXML.setSystemContext(
             new SystemContext("SampleServer"));
58.
59.          SampleServer server = new SampleServer();
60.          java.rmi.Naming.rebind(
             "//localhost/SampleServer", server);
61.          System.out.println(
             "SampleServer bound to registry");
62.       }
63.       catch ( Exception exception ) {
64.          exception.printStackTrace();
65.       }
66.    }
67. }
```

At last—extensible eBusiness

.eXML

Turn everything into eBusiness information

The eBusiness Information Server

Use standards to build eBusiness

eXcelon™

eXcelon is the premier eBusiness information server that uses XML to leverage *all* of the information in your enterprise for eBusiness applications. eXcelon manages and delivers dynamically extensible data to your customers, co-workers, partners and prospects. And, eXcelon's Toolbox lets you get your eBusiness applications up and running *fast.* Information is everywhere. Now you can leverage all of it.

Get Started Today!

Download a free copy of eXcelon.

www.objectdesign.com/excelon

1-800-706-2509

OBJECT DESIGN®

Object Design, Inc. Twenty Five Mall Road, Burlington, MA 01803-4194, email:info@objectdesign.com. Copyright © 1989-1999 by Object Design, Inc. Object Design, ObjectStore, Leadership by Design, ObjectExchange, and the Object Design logo are registered trademarks of Object Design, Inc. Cache-Forward and eXcelon are trademarks of Object Design, Inc. All other trademarks are the property of their respective owners.

explicit steps to preserve it (e.g., by logging its own error message to a file).

Listings 4 and 5 show simple client and server programs that communicate through RMI and use XML to pass detailed error information. Two utility classes, FauxException and ErrorToXML, are referenced in the source code for the client and server programs. These classes are responsible for manipulating the XML documents and provide a convenience layer for Java programmers by hiding low-level XML processing within a more familiar Java interface. The source code for these classes is shown in Listings 7 and 8 (available on *Java Report Online*, www.javareport.com).

Continuing with the examples in Listings 4 and 5, let's use the following scenario:

1. Client locates and invokes the (single) remote method on the server (lines 5–6).
2. The server detects an error during the processing of the remote method (line 48).
3. The server constructs and throws a special remote exception with the converted XML Error (lines 49–52).
4. The client's error-handling logic displays both the RMI RemoteException and the reconstructed server exception (lines 9–14).

The output from the client program is shown in Listing 6.

The difference between the output of the two error-handling routines shows how valuable context information can be passed back to calling programs to better diagnose errors. In this case, the XML Error document used to hold this information piggy-backs on the existing RMI remote exception-handling feature. However, once the error information is captured in XML, there are a number of alternative and complementary strategies that can be employed. For example, the server program may choose to log the XML Error locally in place of a less descriptive text message. Powerful parsing tools can be written or scripted to make use of an XML-based error log.

In the client program, more can be done than just printing the raw XML document. For example, in a browser environment where XML and XSL (with scripting) can be married, a cleverly formatted HTML page can be constructed that draws attention to specific fields and lets the user navigate to related help pages.

An architecture might leverage special XML-based tools that support sophisticated error detection, notification, and diagnostic inquiry into XML Error documents fed by both client and server programs. These tools can gather statistics as they monitor the error input stream(s) and provide suitable notification, alerts, and alarms. Such tools need not be tied to a Java runtime environment, and thus are ideally suited for the newer distributed architectures, including n-tiered corporate Intranets, e-commerce Web applications, or business-to-business integration efforts.

Summary

Error handling in distributed applications presents some unique challenges. Preserving and sharing detailed information among components can greatly assist in application debugging and system recovery. One might argue that any error information captured in a program is only as useful as the format used to represent it, and XML provides an excellent vehicle for such a purpose. XML can be used to capture detailed, contextual information to preserve valuable debugging information and is ideal for sharing extensive error information between languages. Using XML for error handling extends beyond Java and can add value in many computing environments. ■

Steven Jones is a Software Architect at ISA Services Inc., Chicago. He can be contacted at srjones12@yahoo.com.

Quantity reprints of this article can be purchased by phone: 717-399-1900 ext. 139 or by email: sales@rmsreprints.com.

LISTING 6.

SampleClient.

```
D:\>java SampleClient
XMLException: This is an example exception
  at sun.rmi.transport.StreamRemoteCall.
    exceptionReceivedFromServer(Compiled Code)
  at sun.rmi.transport.StreamRemoteCall.executeCall(
    StreamRemoteCall.java:223)
  at sun.rmi.server.UnicastRef.invoke(Compiled Code)
  at SampleServer_Stub.doIt(Unknown Source)
  at SampleClient.main(SampleClient.java:7)
XMLException: This is an example exception
  at SampleServer.doIt(SampleServer.java:9)
  at java.lang.reflect.Method.invoke(Native Method)
  at sun.rmi.server.UnicastServerRef.dispatch(Compiled Code)
  at sun.rmi.transport.Transport$1.run(Transport.java:126)
  at java.security.AccessController.doPrivileged(Native Method)
  at sun.rmi.transport.Transport.serviceCall(Transport.java:123)
  at sun.rmi.transport.tcp.TCPTransport.handleMessages(
    Compiled Code)
  at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(
    TCPTransport.java:607)
  at java.lang.Thread.run(Thread.java:479)
```

Letters *continued from page 8*

Serialized String Limit

In Mark Davis' article on "Serialization" (Durable APIs, *Java Report*, August 1999, Vol. 4, No. 8), he says that there is a 64K-character limit on serialized Strings. In fact, that is true only if the String being serialized consists of nothing but characters found in 7-bit ASCII. Actually, even then it wouldn't be totally true—the limit would really be 64K-1 characters.

The actual limit that applies in all cases is a maximum of 64K-1 bytes in the UTF-8 encoding of a String. The standard serialization method for Strings transforms each UNICODE character into a 1-, 2-, or 3-byte UTF-8 encoding. The number of bytes required to encode a character is determined by its UNICODE value. Characters in the range [0x0001..0x007F] take up 1 byte, those in the range [0x0080..0x07FF] take up 2 bytes, and the rest take up 3 bytes.

Thus, the suggestion made in the article to divide a String into 64K-character chunks is not valid. To be safe, a writeObject() method could:

- Divide a String into 21845-character chunks [21845 = (64K - 1)/3], or
- Catch the UTFDataFormatException exception (thrown when a too-long String is serialized) and try again with shorter Strings.

Samuel C. Yang
samyang@peoplemoverinc.com

Mobilize Your Applications

JUMPING BEANS

lets you quickly create applications which dynamically jump from computer to computer during execution, in a secure, robust fashion.

Fully exploit the capabilities of Java with Jumping Beans, an intuitive and powerful mobility framework allowing never-before-possible solutions in:

- System management,
- Database access,
- Business process implementation,
- Workflow,
- Data gathering and inventory,
- and much more...

1999 OOP '99 MUNICH
BEST OF SHOW
Best Framework

1998 SIGS Conference JAVA Development
BEST OF SHOW
Best Java Frameworks

FULL-FEATURED EVALUATION COPY AT:

www.JumpingBeans.com

OR CALL 408-738-4616

"Jumping Beans" and the Bouncing Bean Logo are trademarks of Ad Astra Engineering, Inc.
Java, 100% Pure Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries

Working With Java Interfaces & Classes

How to maximize design and code reuse in the face of inheritance

DIRK RIEHLE AND ERICA DUBACH

Mark Fisher / Laughing Stock

OBJECT-ORIENTED MODELING is based on classes and objects. A class is the abstraction of several similar phenomena which are found in one or more domains. Objects are the concrete phenomena themselves.

In Java-based implementations of software systems, classes defined in domain models are typically represented as Java interfaces. These interfaces in turn are implemented by additional Java classes.

It is also possible to represent a domain concept, defined as a class in the domain model, directly as a Java class, without the introduction of an interface. However, this solution has several drawbacks. The major issue is that the interface and implementation become one unit of code (i.e., the class), and clients directly depend on this particular class, which prevents exchanging or choosing it freely. Because systems are destined to evolve, this is likely to lead to problems sooner rather than later. Furthermore, any changes to the class directly effect the client code, which possibly can require re-coding or re-compiling. The use of interfaces helps decouple clients from the classes that implement a particular domain class.

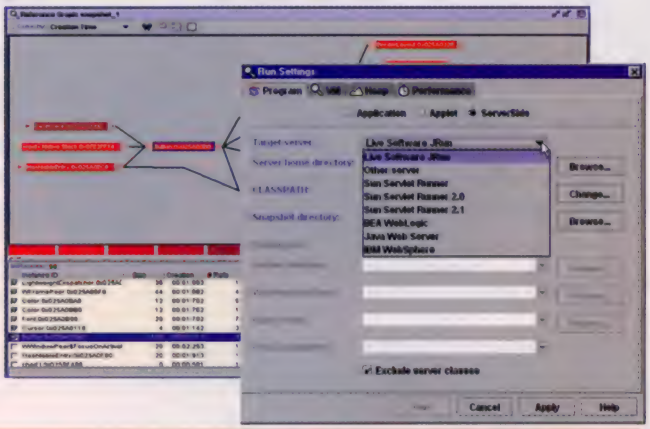
Thus, distinguishing between interfaces and classes that implement them, reduces problems of system change and evolution and can significantly increase flexibility, adaptability, and maintainability.

An earlier article[1] used the implementation of a naming service to illustrate these advantages. A name, such as "net/projects/Geo/ Vroom/" is made up of several name components. Different approaches can be taken to implement the concept of name. The interface `Name` is defined and two example implementations are shown. In one, the `Name` interface is implemented using the class `StringNameImpl`, which maintains the names as a single string with masked separators ("/"). The other implementation is the class `VectorNameImpl`, which maintains the individual name components as strings in a `vector`.

The average urban raccoon can survive on garbage collection alone.

Server-side Java™ performance can't.

You may be thinking that the Garbage Collector eliminates memory problems in Java™. But Java has its own brand of memory "leaks" that result from holding references to objects longer than necessary. While less frequent, these "leaks" are far more severe and can bring your application's performance to a grinding halt. JProbe Profiler's integrated Memory Debugger provides a graphical display of memory usage so you can track down leaks fast and dramatically improve your application's performance.



JProbe Profiler – with Memory Debugger

Debug memory leaks and performance bottlenecks in your Java code

- Debug memory leaks with Reference Graph Browser
- Visualize memory usage patterns with real-time graph
- Pinpoint performance bottlenecks with Call Graph
- Per-line object allocation activity and performance data
- Fully supports JDK 1.1.7 and Java 2
- Windows 95/98/NT and Solaris versions
- Visual Café, Visual Age and JBuilder support

JProbe Threadalyzer

Pinpoint and eliminate deadlocks, stalls, and race conditions

JProbe Coverage

Locate and quantify untested Java code

JPROBE Suite

Available in Developer and ServerSide Editions.

www.klgroup.com/collect

Introducing JProbe ServerSide Edition!

Includes new Web and Application Server Launch Pad:

- IBM WebSphere
- BEA WebLogic
- JRun (Live Software)
- Sun Servlet Runner
- Java Web Server

Now with full Solaris support

KL GROUP
Software Development Productivity

KL Group Inc. Toll-Free: 1-888-361-6205 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

© Copyright 1999 by KL Group Inc. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. JClass, JClass SwingSuite and JClass JarMaster are trademarks of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. JR012

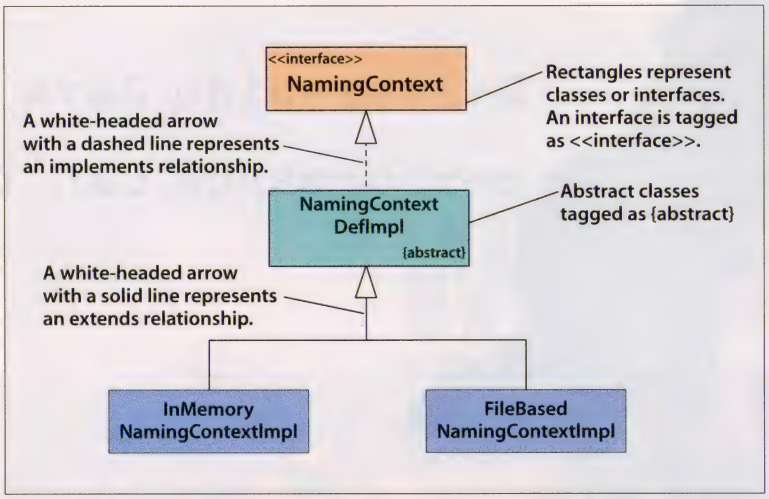

<<interface>>
NamingContext
Rectangles represent classes or interfaces. An interface is tagged as <<interface>>.
A white-headed arrow with a dashed line represents an implements relationship.
NamingContext DefImpl
{abstract}
Abstract classes tagged as {abstract}
A white-headed arrow with a solid line represents an extends relationship.
InMemory NamingContextImpl
FileBased NamingContextImpl

Figure 1. ***The NamingContext interface with two concrete implementation classes and a common abstract superclass.***

We described the tasks of a naming service and discussed two possible implementations.[1] A naming service attaches a name to an object and stores the object under that name in order to retrieve it later, using the given name. The following code describes the naming service as the interface NamingContext, which was taken directly from the CORBA Object Services Specification[2] for the naming service:

```
public interface NamingContext
{
    public void bind(Name name, Object object);
    public void rebind(Name name, Object object);
    public void unbind(Name name);
    public Object resolve(Name name);
    public boolean contains(Name name);
}
```

The methods bind() and resolve() serve to store and retrieve the objects under a certain name. Using the following code, a customer browser object can be stored and retrieved later:

```
// store under name CustomerBrowser
Browser browser = new BrowserImpl();
Name browserName = new StringNameImpl(
  "CustomerBrowser");
LocalServices.getNameService().bind(
  browserName, browser);
...
// retrieve under name CustomerBrowser
Name browserName = new StringNameImpl(
  "CustomerBrowser");
Object object =
  LocalServices.getNameService().resolve(browserName);
Browser browser = (Browser) object;
...
```

We introduced two different implementations of the NamingContext interface, both of which were represented as concrete classes that implement the interface. One is based on a hashtable that resides in memory, which maps names to object references. The other implementation maintains names and object references in one or more files. Storing the information in memory is fast and works well for small and mid-sized numbers of name/object pairs, though the data is lost in the case of a system crash. The file-based implementation, on the other hand, is capable of surviving a crash, because each name/object pair is stored on the hard drive. This second solution is also more suited for larger numbers of objects, but it is usually slower than the memory-resident implementation, at least for the first access, if caching is used.

These two implementations are defined as the InMemoryNamingContextImpl and FileBasedNamingContextImpl classes. The abstract superclass NamingContextDefImpl captures their similarities. The resulting design is shown in Figure 1.

The abstract class NamingContextDefImpl captures common implementation aspects of the NamingContext interface and represents scaffolding for the subclasses. The scaffolding is filled out by these concrete subclasses. The abstract superclass implements the common aspects of its subclasses (DefImpl stands for default implementation) using the concepts of "Factory Method"[3] (p. 107) and "Template Method"[3] (p. 325). The definition of NamingContextDefImpl is shown in Listing 1.

The abstract superclass NamingContextDefImpl has been defined and can now be inherited from. Some functionality has already been implemented, such as the public methods of NamingContext, but not the functionality that needs to vary in the subclasses. The methods doBind, doUnbind, and doContains define the functionality that needs to vary from subclass to subclass (they constitute the so-called narrow inheritance interface). Therefore, they are represented as protected abstract methods of the class NamingContextDefImpl. All code in NamingContextDefImpl delegates subclass-specific functionality to these abstract methods, which are then implemented by concrete subclasses.

The methods doBind, doUnbind, and doContains are implemented using a hashtable for the subclass InMemoryNamingContext, and a file-based implementation for the FileBasedNamingContextImpl class. The art of designing the abstract superclass NamingContextDefImpl consists of finding a suitable inheritance interface, based on which, NamingContextDefImpl can be implemented. Subclasses then only fill out this inheritance interface.

Interface and Class Inheritance

Software systems consist of complex interface hierarchies. We now examine the inheritance of interfaces and how they relate to the concepts of implementation classes and code reuse.

To demonstrate this, we continue the naming service example. An object name, such as http://www.riehle.org/fun/index.html, is a complex construct consisting of several parts, each of which requires elaborate lookup methods. A Web browser first interprets the protocol to use, then the Internet Domain Name Service supplies the IP address for "www.riehle.org," and finally an http server delivers the content of fun/index.html by reading a file for the given file name. The file name interpretation in the local file system might be complex as well, e.g., when symbolic links use NFS–Mount–Points to refer to files across file systems.

It is desirable to separate those parts of a name which require complex implementations for resolving them. Ideally, the handling of the separated parts can be changed without affecting the other parts.

The Composite design pattern[3] (p. 163) reviews how to do

WHO SAYS THERE ARE NO ENTERPRISE JAVA™ APPLICATIONS?

JAVA™ Based Sales Force Automation Software

Give Your Sales Force The Power of The WEB!

Industry Leaders Who Have Applied Sales Vision

- Qwest
- Reed Publishing
- Scudder Kemper
- McGraw-Hill
- Norwest Mortgage
- Northern Trust
- US West
- Harcourt-Brace
- Merck

Jsales™

Jsales is the first sales and marketing solution to fully leverage the power of the Internet. Multi-billion dollar companies like *McGraw-Hill* and *Norwest Mortgage* are using Jsales to improve sales productivity, customer relationships, forecasting, and partner communications. Why did these firms select Jsales? Jsales is the **ONLY** SFA solution available based entirely in Java and is also the only solution that allows you to use **YOUR** data/business model. A unique component architecture allows you to maintain your investment in your current data model and, we'll prove it!

Jsales advantages over traditional applications:

- Entire application deployment over the Internet
- Three-tier architecture eases central application administration
- Runs on any platform
- Ideal linkage tool for sales channel partners
- Source code provided
- Completely adaptive component architecture supports any business-selling or data-model
- Totally open, no proprietary tools, any Java-based IDE
- Complete enterprise application functionality

SALES VISION
SALES VISION
www.salesvision.com

Call today to apply for a FREE customized prototype based on YOUR business

1-800-275-4314

© 1999 Sales Vision

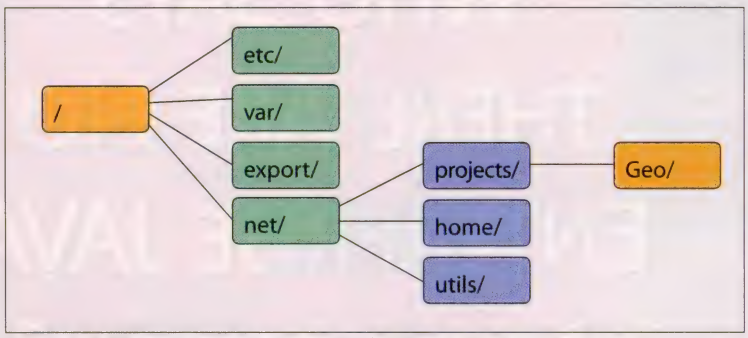


Figure 2. ***A file system as an example of a hierarchical name space.***

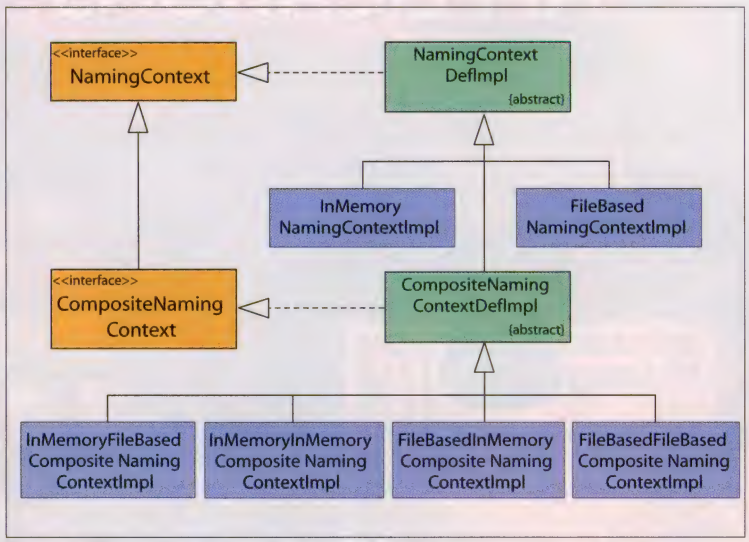


Figure 3. ***Illustration of interface and default implementation inheritance.***

so. It describes how to build recursive object hierarchies ideal for hierarchical name spaces.[4] A hierarchical file system is a good example, where each folder defines the name space for the files it contains. From each folder, the files can be accessed without having to provide the full path name. Figure 2 shows a simple file tree.

The file named "/net/projects/Geo/Geo.proj" can be referenced from the so-called naming context "/" using the name "net/projects/Geo/ Geo.proj," or from "/net/" using the name "projects/Geo/Geo.proj," etc. An object with a name ending with the "/" separator is a so-called naming context, which can take a given name and return the appropriate object—usually a file, directory, or symbolic link. The interface for such naming contexts is defined as NamingContext.

To resolve the name "net/projects/Geo/ Geo.proj," it makes sense that the naming context "/" does not do all the work. Rather, "/" should know its subcontexts and delegate the work to them, because they themselves are naming contexts. Subcontexts of "/" from the above example are "etc/," "var/," "export/," and "net/." To resolve the name of a file, the naming context first removes the part of the name that leads to the selected subcontext. Therefore, "/" removes "net/" from the name "net/projects/Geo/Geo.proj," leaving "projects/Geo/Geo.proj." The shortened name gets passed to the subcontext object, which in turn goes through the same process until the searched-for file "Geo.proj" is found. A search down the tree structure is the result from the initial name.

In real applications, naming contexts do not map onto directories one-on-one, typically because of efficiency reasons. A reasonable solution, for example, represents each file system with a naming context object that is attached to an embedding super-ordinate naming context, where it is mounted.

To add subcontexts to embedding context objects, the interface NamingContext needs to be extended. This leads to the interface CompositeNamingContext, which extends NamingContext:

```
public interface CompositeNamingContext
    extends NamingContext
{
    public void bindSubContext(
      Name name, NamingContext nc);
    public void rebindSubContext(
      Name name, NamingContext nc);
    public void unbindSubContext(Name name);
    public NamingContext resolveSubContext(
      Name name);
    public boolean containsSubContext(Name name);
    public Vector getSubContexts();
}
```

An object that implements the CompositeNamingContext interface is an object that first of all functions as a normal naming context object, because it inherits its definition from NamingContext. In addition, parts of the name space it is responsible for can be attached as subcontexts. This is supported by methods for the management of child-objects in a tree structure, like getSubContext(), bindSubContext(), etc. The interface CompositeNamingContext is derived from the COSS specification for naming services.[2]

A composite naming context object delegates the search for an object to one of its subcontexts when the object name indicates that the subcontext is responsible for it.

An implementation for CompositeNamingContext must therefore provide additional functionality to the NamingContext implementation. First, it must be able to manage subcontexts as part of its name space; and secondly, it must adjust the NamingContext functionality to function properly in this extended-use scenario.

- The implementations of the methods bindSubContext, rebindSubContext, etc., are similar to the NamingContext methods bind, rebind, etc. Typically, the subcontext might be searched for in a memory-resident hashtable to see if it is part of the local name space. Sometimes, however, a file-based solution might be preferable for some implementations as just discussed.
- Additionally, the functions bind, rebind, etc., must be adjusted. A check must be performed before each call to these functions as defined in NamingContext to see if the name to be resolved is in the name space directly managed by the current naming context object, or whether it must be passed on to a subcontext.

Let's consider a few possible implementations before looking at specific code examples. The inheritance hierarchy might be different depending on the kind of reuse desired.

Let us assume that the class InMemoryCompositeNamingContextImpl implements the memory-resident solution to manage the subcontexts. Should the class then inherit from InMemoryNamingContextImpl? If so, the resulting implementa-

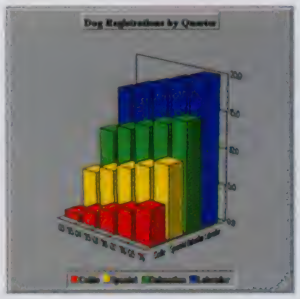
StudioJ

JavaBeans™ components and classes for developing great-looking Java™ applications

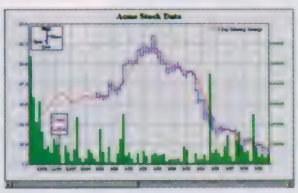
The Hottest Java Tools for Data Visualization

Rogue Wave's StudioJ offers high-quality, integrated, pure Java components for visualizing

enterprise data. Our data interface allows you to access data from a variety of sources, such as CORBA, JDBC, TCP/IP, etc. You can easily read in the data, populate the unified data model and then quickly view data by using our layout template to easily map



to complex overlay charts or incongruous and various-sized grid cells (without trashing your layout). Our new release includes features for visualizing XML data; an ORB data channel for viewing data from anywhere in the enterprise; and an extensible model-view-controller architecture. Other features include GIF exportation, support for customizing applets using HTML tags (no Java code required!), JFC support, plus Java 2/JIT compatibility. StudioJ makes a great add-on to any Java IDE environment, including JBuilder, Visual Café, and Visual Age for Java. StudioJ includes full source code and JavaBeans.



StudioJ is the solution for **Enterprise Data Display**.

Check out our FREE demos at
www.roguewave.com/ad/studioj
or call 800-487-3217 and mention this ad.

Rogue Wave®
SOFTWARE
Components Without Limits

Rogue Wave: 800-487-3217 • D.A.CH.: +49-6103-59 34-0 • UK: +44-118-988 0224
France: +33-1-4196 2626 • Italy: +39-02-3809 3288 • Rest of Europe: +31-20-301 26 26

Rogue Wave is a registered trademark, and StudioJ is a trademark of Rogue Wave Software, Inc. Java and all Java-related trademarks are registered trademarks of Sun Microsystems, Inc. All other trademarks are the property of Rogue Wave Software or their respective holders.

www.roguewave.com

TOPLink does more.

The world's most advanced object/relational mapping tools keep getting better!

TOPLink is a trademark of The Object People, Inc. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other brand and product names are trademarks or registered trademarks of their respective owners.

TOPLink™
THE OBJECT PEOPLE

Available from The Object People

TOPLink speeds Java application development

TOPLink is a product line of advanced object/relational mapping tools that allow you to map complex business objects and EJBs to relational databases so you can leverage your investment in data *and* deploy new applications with Java.

With TOPLink, you can save as much as 40 percent of the time you would otherwise spend getting your Java application up and running.

TOPLink incorporates a meta-data architecture that separates the details of persistence from the business logic. No SQL programming is required. TOPLink supports any JDBC-compliant database including DB2, Oracle, Sybase, Informix and SQL Server (also new JDBC 2.0 support), as well as non-relational data sources.

TOPLink for Java 2.0 comes with these great features:

- Visual Builder tool for creating mappings
- Flexibility with 15 types of mappings
- Object-level transactions
- Queries in terms of objects, not SQL
- Transparent 3-tier support to provide remote session and unit-of-work functionality
- Java 2 support for collections and weak referenced caches
- Optimistic and pessimistic locking
- Object caching and identity
- Batch reading and writing
- Support for Oracle8*i*
- Support for **Enterprise JavaBean** persistence, including:
  - Bean-managed persistence
  - Transparent mapping of EJBs to relational databases
  - Fully integrated functionality with Session beans
  - JTS integration

TOPLink for BEA WebLogic provides application server support.

TOPLink for BEA WebLogic builds on the existing EJB support and mature feature set of TOPLink for Java, and provides fully integrated object/relational mapping for container-managed Entity bean persistence with the BEA WebLogic application server.

For a **FREE** 30-day evaluation copy of TOPLink for Java, or more information, contact us by e-mail at info@objectpeople.com, phone (613) 225-8812, or visit our web site.

Contact us to enquire about other EJB server support.

www.objectpeople.com

CANADA • U.S.A. • U.K. • GERMANY

THE OBJECT PEOPLE
www.objectpeople.com

tion resolves the name as well as maintains the subcontexts in memory.

On the other hand, if an implementation is chosen that manages names using a database and where the subcontexts are in files, then the class inherits from FileBasedNamingContextImpl. A different combination needs to be chosen if the name table is to be kept in memory and the subcontexts stored as files.

Concrete classes, such as InMemoryNamingContextImpl are "semantically closed," meaning that their implementation determines their entire behavior. A further implementation consideration is that any subclass of such a semantically closed class is likely to contradict the inherited implementation, a phenomenon described as part of the abstract superclass rule.[5]

A solution to this problem is presented below. It takes into account that inheriting from concrete classes is usually not a good idea because the implementation semantics might get confused and an explosion of implementation variations might be produced.

Figure 3 shows the suggested class hierarchy that uses the idea of abstract default implementations just introduced.

Here, again, an abstract default implementation class exists with the name CompositeNamingContextDefImpl, which inherits from NamingContextDefImpl. Just like the NamingContextDefImpl class, this implementation uses a narrow inheritance interface, as seen in Listing 2.

All these functions are parallel to NamingContextDefImpl. The main part of the implementation is based on a small inherited interface, requiring the subclasses only to fill out the inherited interface.

The bind, rebind, etc., methods need to be adjusted, of course. This is done using a helper method that determines if a name is in the current name space or within a registered subcontext:

```
// belongs to CompositeNamingContextImpl
protected NamingContext findSubContext(
  Name name)
{
    for ( int i = 0; i < name.noOfComponents(); i++ )
    {
        Name context = name.context(i);
        if ( containsSubContext(context) )
        {
            return resolveSubContext(context);
        }
    }
    return null;
}
```

Using this helper method, the function resolve() can determine if the name to be resolved is maintained locally or whether the task needs to be passed to a subcontext:

```
// belongs to CompositeNamingContextDefImpl
public Object resolve(Name name)
{
    NamingContext subContext =
      findSubContext(name);
    if ( subContext != null )
    {
        Name subName =
          name.without(subContext.getName());
        return subContext.resolve(subName);
```

LISTING 1.

The definition of NamingContextDefImpl.

```
public abstract class NamingContextDefImpl
  implements NamingContext
{

public void bind(Name name, Object object)
{
  if ( ! contains(name) ) doBind(name, object);
}

public void rebind(Name name, Object object)
{
  if ( ! contains(name) ) return;
  doUnbind(name);
  doBind(name, object);
}

... and the other methods

// this is the inheritance interface
protected abstract void doBind(
  Name name, Object object);
protected abstract void doUnbind(Name name);
protected abstract void doContains(Name name);

// that is all! no further implementation state which
// would (possibly unnecessarily) burden the subclasses

}
```

LISTING 2.

CompositeNamingContextDefImpl uses a narrow inheritance interface.

```
public abstract class CompositeNamingContextDefImpl
  extends NamingContextDefImpl
  implements CompositeNamingContext
{

... constructors and initialization code

public void bindSubContext(Name name, NamingContext nc)
{
  if ( ! contains(name) ) doBindSubContext(name, nc);
}

public void rebindSubContext(Name name, NamingContext nc)
{
  if ( ! contains(name) ) return;
  doUnbind(name);
  doBind(name, nc);
}

... and the other methods

// here the inheritance interface
protected abstract void doBindSubContext(Name name, NamingContext nc);
protected abstract void doUnbindSubContext(Name name);
protected abstract void doContainsSubContext(Name name);

}
```

Access Data... Fast

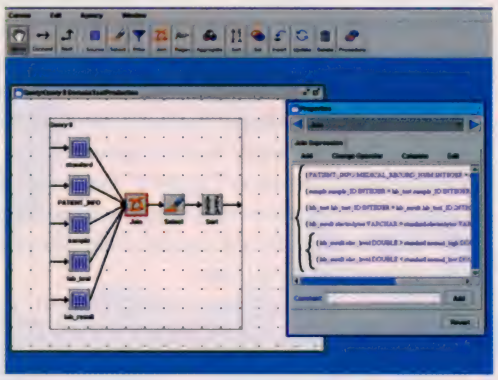
With Cerebellum™, New Applications Can Get Off the Ground at Lightning Speed.

Get out of the slow lane and bypass development gridlock caused by:

- Time-consuming coding and numerous SQL dialects;
- Multiple, incompatible data sources;
- Users constantly changing their minds;
- Only 24 hours in a day.

It's time to speed things up...

Cerebellum is an application development platform, written in **Java 2 with CORBA technology**, which gives you the freedom to build applications that access multiple, distributed data sources. Cerebellum's single interface lets you access, integrate and manage data from any source in a matter of seconds. Queries are created visually with our drag-and-drop graphical query language — eliminating the need to write SQL code. And with Cerebellum, queries that access one database can access another database without any redesign — it's as easy as changing your mind.



Test drive Cerebellum with a **FREE** 30-day trial and leave the development roadblocks behind.

To download, visit

www.cerebellumsoft.com

Toll free: (888) 862-9898

Cerebellum SOFTWARE

Declare Data Independence

Cerebellum is a trademark of Cerebellum Software, Inc.
Java is a trademark of Sun Microsystems, Inc.

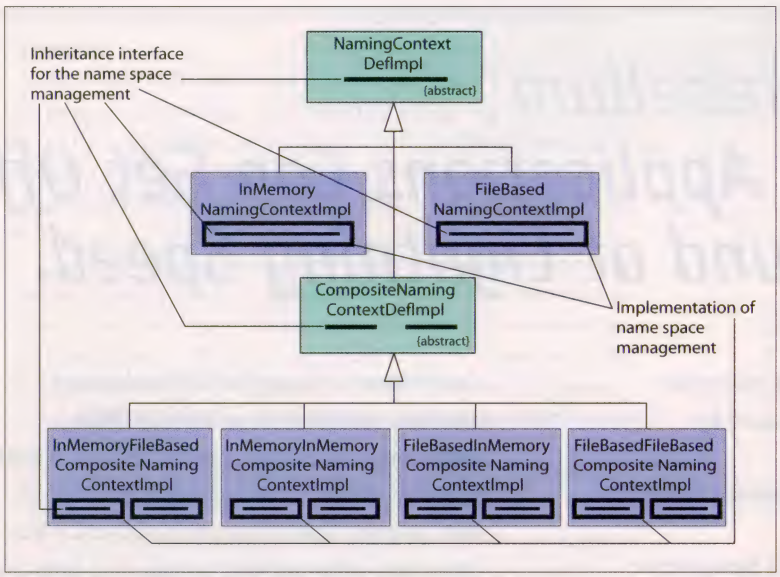


Figure 4. ***Illustration of the inheritance interfaces of abstract superclasses.***

```
    }
    else
    {
        return super.resolve(name);
    }
}
```

The concrete subclasses of CompositeNamingContextDefImpl only need to implement the unfinished scaffolding parts. Those are the methods from the inheritance interface of NamingContextDefImpl for the name management, and the inheritance interface of CompositeNamingContextDefImpl for the subcontext management. Because there are two possible implementations for each, four combinations of subclasses are possible, as is shown in Figure 3. This combinatorial explosion can be counteracted using the implementation factoring technique detailed in the next section, though the principle is discussed here.

For each interface an abstract default implementation class exists that runs parallel to the interface hierarchy but that is not instantiable. The more complex the hierarchy is, the more inheritance interfaces exist that must be implemented by the subclasses. By using the concept of default implementation, a certain degree of direct code reuse can be achieved, which is particularly good in the face of the Java limitation that classes can only inherit from a single superclass.

If the concept represented by an interface is to be instantiable, then concrete subclasses of the corresponding abstract default implementation class should exist that can be used directly by the clients.

The next question to ask is what happens when the interface hierarchy is not based on simple inheritance but uses multiple inheritance. This is possible though the motivation for using multiple inheritance should be inspected first. From our experience, domain-specific modeling usually produces a simple inheritance structure. Multiple inheritance is only useful for attaching additional protocols to a single-inheritance tree. These protocol interfaces are recognizable by their ending "-able", e.g., Cloneable, Serializable. They usually do not have their own default implementation, but if they do, the concepts discussed in the next section are very helpful.

Factoring Implementations

As the previous section shows, complex objects often consist of different aspects that can easily be separated in the implementation. A composite naming context needs to manage its name space as well as its subcontexts. These tasks both happen in the same object, yet the implementations of them are not necessarily dependent on each other.

The introduction of an abstract superclass, which implements the standard behavior of an interface, leads to the concept of an inheritance interface. An abstract superclass defines a set of protected class-hierarchy internal methods, the inheritance interface, in such a way that the superclass' own implementation can make use of these methods without having to implement them. Figure 4 uses the ongoing example and shows the inheritance interface for the management of the name space and where it is implemented. The bold lines represent the definition of an inheritance interface in a class, and a bold line with a box indicates the implementation.

The CompositeNamingContextDefImpl level adds the inheritance interface for the management of subcontexts, which results in an explosion of possible subclasses. These are the classes InMemoryFileBased-, InMemoryInMemoryFileBasedInMemory-, and FileBasedFileBasedcompositeNamingContextImpl. (Note: These horribly long names will be used throughout for the illustration of our example. For the implementation, however, shorter names would be preferable or the long names could be hidden behind convenience methods).

The prefixes InMemory and FileBased indicate the chosen implementation for the management of the name space and the subcontexts. The two classes InMemoryFileBased- and InMemoryInMemory-CompositeNamingContextImpl both implement an in-memory name space management but use different mechanisms to implement the subcontext management. They cannot inherit from a common superclass, though (e.g., InMemoryNamingContextImpl), without losing other advantages. The main advantage in this example is the superclass CompositeNamingContextDefImpl that defines the standard behavior for composite naming contexts. Of course, it is possible to choose a different superclass, though this would result in conflicts for other combinations, which is a result of the limitation that classes only have single inheritance. No matter how much effort is put into the design of a class hierarchy, optimal code reuse is not possible in Java using only inheritance.

How can we avoid the implementation redundancy and the resulting hassles with maintenance and enhancements? By using object composition instead of inheritance for code reuse. The most important enabler for this is the concept of inheritance interfaces, which we have already introduced.

The implementation of the inheritance interface is passed to a new object, instead of implementing it in every subclass. This new object is an example of a class which implements this (and only this) interface. Because the objects might differ, the implementations can differ as well, as long they sat-

Feeling Exposed?

Don't Reveal Your Source...

JCloak - Code Obfuscation That Doesn't Byte.

jcloak™

force5.com

Force5 software

Force 5 and JCloak are trademarks of Force 5 Software, Inc. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

sfy the inheritance interface.

The new subclass NamingContextFrontImpl of NamingContextDefImpl implements the inheritance interface by delegating the function calls to those objects that implement the interface NamingContextBackend. For our example, these could be the classes InMemoryNamingContextBackImpl or FileBasedNamingContextBackImpl.

Another class, CompositeNamingContext FrontImpl, delegates the implementation of the name space management and the subcontexts to other objects. For each part of the implementation there is a "Backend" class, which fulfills just this aspect and whose objects provide the implementation for the "Frontend" classes.

The implementation of the inheritance interface for NamingContextFrontImpl is shown in Listing 3. The following code produces a naming context object.

```
// selecting the implementation
NamingContextBackend ncBackend =
  new InMemoryNamingContextBackImpl();
// creating the naming context object
NamingContext nc =
  new NamingContextFrontImpl(ncBackend);
```

In the case of the composite naming context, the code is similar, except that it is parameterized with two implementation objects (Listing 4—see *Java Report Online's* code section). If a certain combination of inheritance objects is particularly common, that combination can be captured with a separate conveniece subclass. The combination of a file-based name space management and a file-based subcontext management would lead to the following definition:

```
public
  class FileBasedInMemoryCompositeNaming
    ContextFrontImpl
     extends CompositeNamingContextFrontImpl
{

public
  FileBasedInMemoryCompositeNamingContextFrontImpl()
{
    NamingContextBackend ncBackend =
        new InMemoryNamingContextBackImpl();
    CompositeNamingContextBackend cncBackend =
        new FileBasedNamingContextBackImpl();
    initialize(ncBackend, cncBackend);
}

}
```

A client can then use this well-defined class to create an object directly without having to write the configuration code.

The principle described here of delegating implementations to separate objects, corresponds to the design pattern Bridge[3] (p. 151). This design pattern describes how an abstraction (Frontend, e.g., NamingContext) is parameterized with an implementation (Backend, e.g., NamingContext Backend). The implementation object could also be considered a Strategy, another design pattern[3] (p. 315). However, because strategies are often meant to change during runtime and this is not an important requirement for this example, Strategy is not used here. ■

LISTING 3.

Implementation of the inheritance interface for NamingContextFrontImpl.

```
public class NamingContextFrontImpl
  extends NamingContextDefImpl
{

// implementation state
protected NamingContextBackend
  fNamespaceBackend;

public NamingContextFrontImpl(
  NamingContextBackend ncBackend)
{
  fNamespaceBackend = ncBackend;
}

protected void doBindSubContext(
  Name name, NamingContext namingContext)
{
  fNamespaceBackend.bindSubContext(
  name, namingContext);
}

protected void doUnbindSubContext(Name name);
{
  fNamespaceBackend.unbindSubContext(name)
}

protected boolean doContainsSubContext(
  Name name);
{
  return fNamespaceBackend.containsSubContext(name);
}

}
```

References

1. Riehle, D. and E. Dubach, "Working with Java Interfaces and Classes: How to Separate Interfaces From Implementations," *Java Report*, Vol. 4, No. 7, July 1999.
2. OMG, *CORBA Object Services Specification*, Object Management Group, 1997.
3. Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
4. Mullender, S., *Distributed Systems*, Addison-Wesley, 1992.
5. Hürsch, W. L., "Should Superclasses be Abstract?" In Proceedings of *the 1994 European Conference on Object-Oriented Programming (ECOOP '94, LNCS 821)*, Edited by Mario Tokoro and Remo Pareschi, Springer–Verlag, 1994, pp. 12–31.
6. Riehle, D., "Arbeiten mit Java-Schnittstellen und-Klassen," *Java Spektrum*, Sept./Oct. 1997, pp. 26–33.

Dirk Riehle works as a software engineer at Credit Suisse in Zurich, Switzerland. He can be contacted at riehle@acm.org.

Erica Dubach works in the New Technology department of Atraxis AG (SAir Group) in Zurich, Switzerland. She can be contacted at dubach@acm.org.

Quantity reprints of this article can be purchased by phone: 717-399-1900 ext. 139 or by email: sales@rmsreprints.com.

THE BEST OF
LIVE AND COMPUTER
BASED TRAINING

View the material anytime and at any pace

Direct phone access to Java experts

Peer-to-Peer IDE for concurrent Teacher/ Student programming

Other Benefits

Cost/Time Efficient

More individual attention than traditional classroom or cd-based training

Follow-up help available after course completion

For a limited time get a **free CD** of Gerry Seidman's lecture on Java Technology and Integrating Java Technology into your Enterprise. To receive your CD, visit our website at www.iam-training.com for more information.

IAM Consulting
200 West 72nd Street, Ste 35
New York, NY 10023
(212) 580-2700
www.iam-training.com

Learn or Improve your Java™ Skills

Unique Distance Training Format

Affordably train 1–10–100

At the office or at home

By one of the top Java technologists

At your own pace and schedule

Not just syntax
we show you how to produce

Clear
Clean
Maintainable Systems

That Run
and...
Produce them in a reasonable amount of time

Now for the first time
courses by Gerry Seidman,
Java expert and pioneer
are widely available

IAM Courses Include

Java For Programmers

This applied OO/AD approach to learning Java is ideal for programmers with any background. In addition to its comprehensive coverage of the Java language and libraries, you will learn how to design and implement two and three tier-client/server and Web-based applications.

Expert Java: Designing Enterprise Systems

Designing scalable, secure, distributed applications starts with an understanding of Java internals. From there, learn the technology and design issues behind sophisticated applications using Distributed Objects (RMI/CORBA), framework architectures (Java Beans/Servlets), Legacy Systems (JDBC/JNI), advanced Networking/Multithreading techniques (building resource pooled servers) and much more.

EJB and Transaction Based Systems

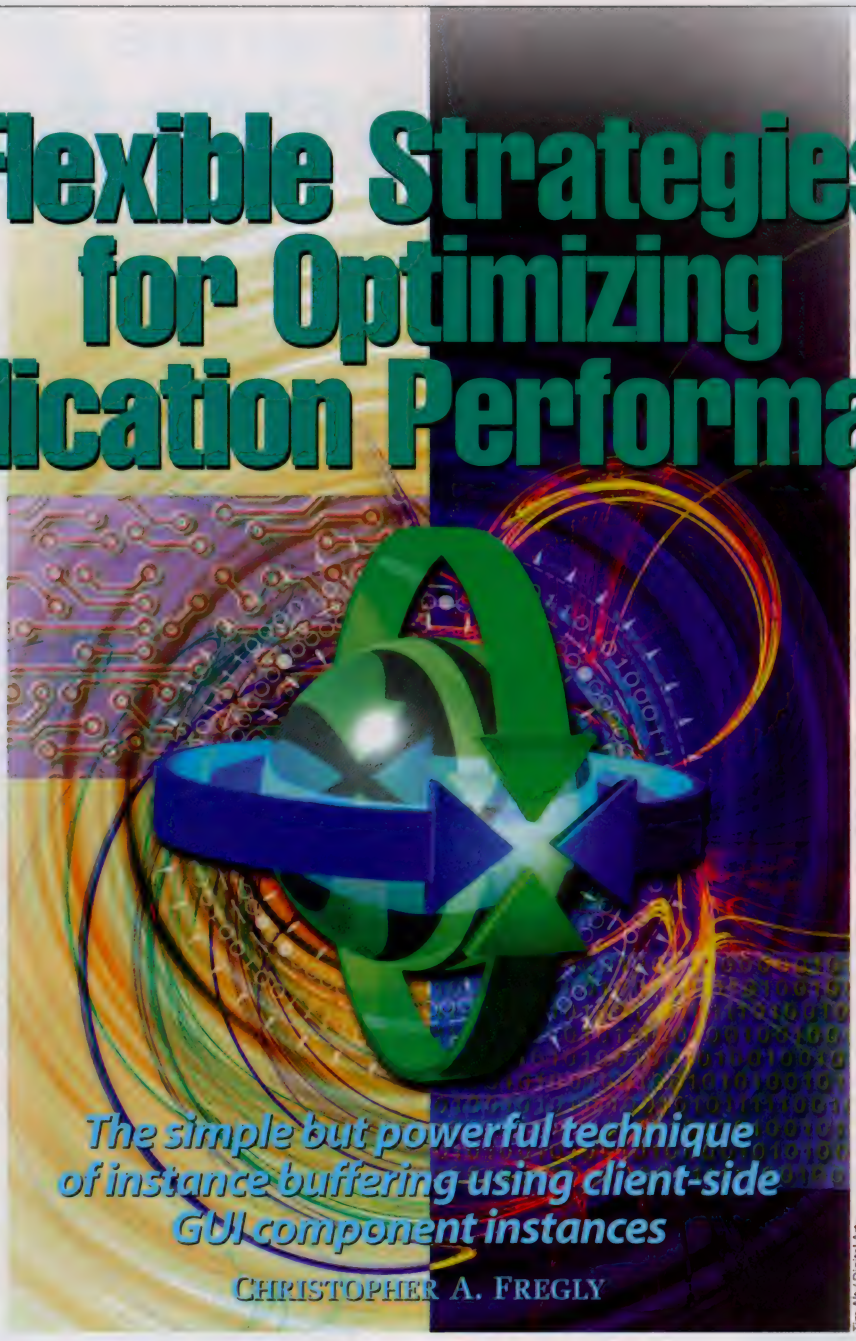
Learn how to rapidly design, develop and deploy enterprise wide, server-side business solutions in Java using EJB. Included is a comprehensive coverage of the Java Transaction API (JTS), Java Naming and Directory Services (JNDI), RMI and RMI/IIOP, bundling and deploying EJBs in real application servers as well as practical design techniques for using EJBs in applets, applications and servlets.

Training and consulting at your site is also available.

Java is a registered trademark of Sun Microsystems.

iam

Flexible Strategies for Optimizing Application Performance



The simple but powerful technique of instance buffering using client-side GUI component instances

CHRISTOPHER A. FREGLY

Tim Alt / Digital Art

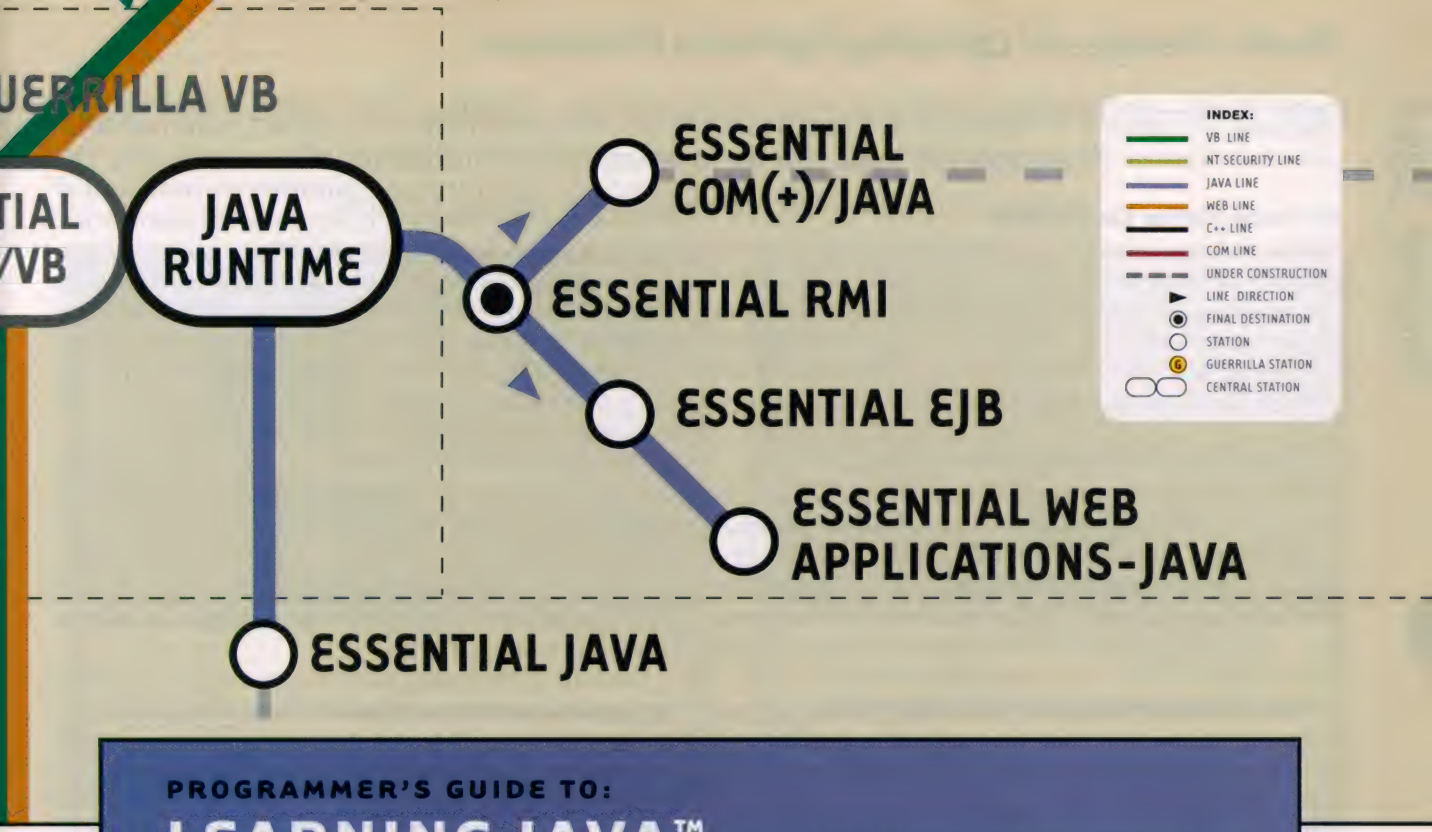
APPLICATION COMPLEXITY AND functionality increase, application performance decreases proportionally. Full-featured GUI applications demand additional resources as power users demand additional features. It is often difficult to maintain acceptable start and response time given additional complexity and functionality. The simple, short-term solution for improving application performance is to upgrade the hardware involved. The better, long-term solution is to optimize the application design and implementation. By maximizing the use of all available resources, the application can more easily adapt to changes in deployment environment, complexity, and functionality.

Additional functionality directly translates to more panels, text fields, buttons, and other widgets common in a GUI. Additional resources are required to load, create, and manipulate these additional features. Limited memory, slow network and processor speed, and user impatience discourage the developer from loading, constructing, and initializing every GUI component on application startup.

Instance buffering, similar to double buffering, maximizes the performance of any well-designed Java application by loading, constructing, and initializing an instance before it is needed. Both buffering techniques improve the effective response time, at the expense of consuming other resources such as memory and processor utilization.

Because resource scarcity is most common on the client side, I'll explain the simple but powerful technique of instance buffering using client-side GUI component instances. This technique is not limited to the client side, however. Instance buffering is useful in optimizing server-side instances, including CORBA and RMI remote services.

Design alternatives are analyzed with respect to buffering: 1) which instances, 2) at what point during runtime, and 3) to what level—where "level" is defined as class loaded, in-



PROGRAMMER'S GUIDE TO:

LEARNING JAVA™ for distributed computing

FASTEST TRACKS

ESSENTIAL JAVA

This course is for professional developers currently working in another language who want a fast-paced and in-depth introduction into using Java so that they can create stand-alone applications immediately.

INTENSIVE JAVA

This course is about getting inside the Java virtual machine to better understand its expectations, logic, design and resulting behavior with your application. You'll learn many undocumented tricks and techniques that the VM uses and how you can surmount any optimization, deployment, and security problems in your Java application.

ESSENTIAL RMI

This course is about the technology behind building distributed systems with Java and the core RMI features in the JDK. You will be provided with a real understanding of how to get each feature to work for you and the strengths and weaknesses of each.

TRANSACTIONAL PROGRAMMING WITH EJB

This course will show you the programming model defined by the EJB's specification and builds on your knowledge of Java RMI.

ESSENTIAL WEB APPS-JAVA

This course focuses on writing distributed applications written in Java that use browser-based clients. You'll learn about Java applets and servlets and how to use Beans for managing data.

NO TOKENS
WHAT THE TOUR BUSES DON'T SHOW YOU

Java Time Table
hands-on courses underground

Java Stops	Time		Location
	October	November	
Essential Java for developers	10/04-10/08		Los Angeles
			Boston
		11/01-11/05	Portland
			England
Intensive Java the runtime environment	10/25-10/29		Los Angeles
			Boston
			Portland
			England
Essential RMI Java distributed objects	OFFERED IN DECEMBER		Los Angeles
			Boston
			Portland
			England
Transactional Programming with EJB	10/20-10/22		Los Angeles
			Boston
			Portland
		11/03-11/05	England
Essential Web Apps - Java for Java technologies	10/13-10/15		Los Angeles
			Boston
			Portland
		11/17-11/19	England

Training Stations:

Los Angeles
Boston
Portland, OR
England

Registration and Information:

800.699.1932
for the US

08000.562.265
within the UK

+44 1242 525 108
within Europe

DEVELOPMENTOR℠
www.develop.com

Java is a registered trademark of Sun Microsystems.

stantiated, or initialized. In addition, UML diagrams and a Java implementation are provided.

Instance Buffer Description

Double buffering is a widely used technique in the software graphics world to increase the effective performance of image rendering. Double buffering improves the effective or perceived performance by rendering images off-screen (in the background) before they are viewed. Swing contains API support for double buffering in a key Swing base class:

```
JComponent.setDoubleBuffered( boolean aFlag )
```

Double buffering should be used anytime resources such as memory, network speed, and processor speed are plentiful.

In addition, AWT offers the MediaTracker utility class, designed to asynchronously load images, audio, and video in the background, allowing other threads to continue execution. This multithreaded behavior greatly improves application response time when used appropriately.[5]

Similar to double buffering and MediaTracker, instance buffering creates complete instances of Java classes as quickly as possible when needed. A "complete instance" is defined by the following three levels:

1. The class (byte code) is downloaded, verified, and loaded into the VM.
2. The instance is constructed and memory is allocated on the heap.
3. The instance is initialized and ready for processing.

The flexibility gained by decoupling the loading, constructing, and initializing of any instance enables significant improvement in the performance of a GUI application.

Instance buffering will improve performance in any properly designed and properly partitioned application. A properly designed system (suitable for instance buffering) is characterized by grouping related instances into logical units to be buffered together. The best GUI example is a tab panel group that visually associates related tab panels according to the business domain. A properly implemented system decouples any resource-intensive initialization code—such as the initialization of a large combo box list of thousands of countries—from the constructor of the object.

To provide the granular flexibility of instance buffering, a thin layer is introduced around each participating instance. This layer—the instance buffer—is responsible for loading the class, constructing the instance, and initializing the object. This abstraction provides the flexibility needed to buffer instances when most optimal during runtime.

Instance buffering provides the developer with the level of granularity needed to take an instance from class loading to instance construction and instance initialization as efficiently as possible. Design tradeoffs for choosing the level of buffering with respect to available memory, network speed, and processor speed are discussed later.

What to Buffer?

Any instance can be buffered, including a large list of countries, a tab panel, or even a CORBA service. This technique can improve performance on any tier, provided the appropriate instances are chosen for buffering.

An appropriate software development methodology, such as Rational's Unified Process, will create functional components or "units of work" within an application. The granularity of each of these components is usually a set of activities that could be visually represented by a small number of tab panels within a tab group.

The challenge is to find the right number of related instances to improve effective response time of the overall application. Buffering too many instances will increase the component instance's overall startup time. This will most likely lose the user's interest. Buffering too few instances will render the application sluggish and unresponsive.

The developer must carefully choose which instances will offer the best improvement in performance, without negatively impacting start time within the limits of available resources such as memory, network speed, and processor speed. As a rule of thumb, the developer should buffer as many frequently used component instances as acceptable on application startup.

Consider the following example: Most common brokerage applications allow a customer to check account balances, transfer funds, and trade securities. Our application is partitioned into the following three functional components:

- AccountMaintenanceComponent
- AccountTransferComponent
- SecurityTradeComponent

LISTING 1.

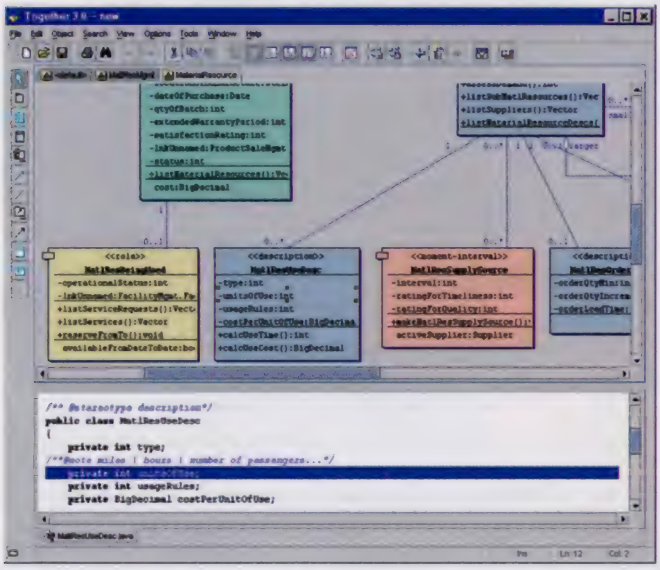
Using BufferFacility.

```
//Create new BufferFacility
//with PayWhenConvenientBufferStrategy
BufferFacility bufferFacility = new BufferFacility(
new PayWhenConvenientBufferStrategy() );

//Create unique String identifier for this instance
String id1 = "SamplePanel1";

//Add the class name
//"com.isaserv.ui.SamplePanel" to
//the BufferFacility for buffering
bufferFacility.add( id1, "com.isaserv.ui.SamplePanel",
null, new Object[0] );

//Add a BufferListener to receive
//BufferState change notifications
bufferFacility.addBufferListener( id1,
new BufferListener(){
public
void stateChanged( BufferEvent evt ){
System.out.println(
evt.toString() );
}
} );

//Start buffering instance identified
//by id1 completely to the
//BufferState.INITIALIZED state
bufferFacility.startBuffer( id1, BufferState.INITIALIZED );

//Synchronously retrieve the fully
//initialized instance identified by id1
SamplePanel panel =                    (
 SamplePanel)bufferFacility.getInstance( id1 );
```

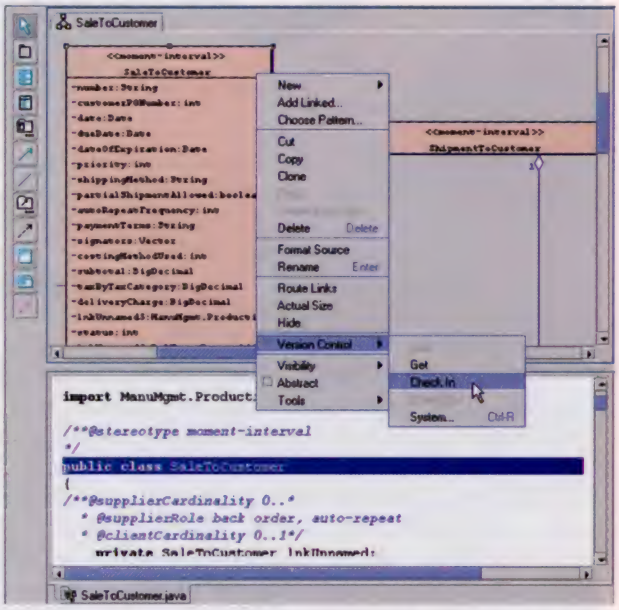
New!

Together/J's EJBs, GoF patterns, and simultaneous round-trip engineering—what Home Depot calls "the backbone of all our Java development"

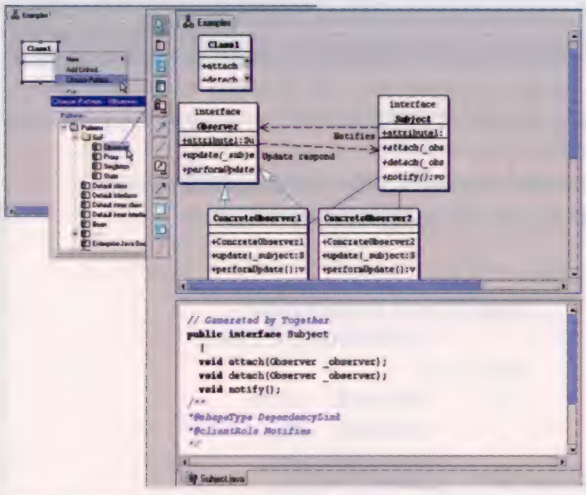
Download the free TJ Whiteboard and experience TJ for yourself



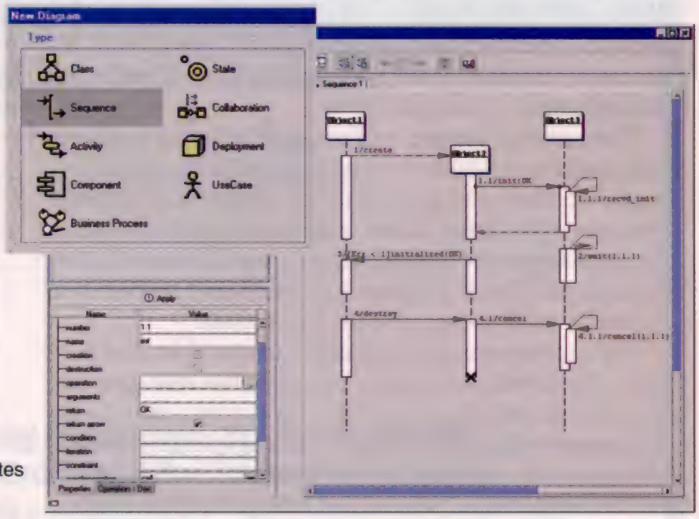
Simultaneous round-trip—always in-sync always up-to-date, complete freedom to work visually or textually as you see fit



Version-control features are a just mouse-click away—complete end-to-end versioning of both models and source across your entire team



Select a class, choose EJB™ or a GoF pattern, and TJ auto-generates it for you – in Java or C++ (or both, in Together/Enterprise)



UML 1.3 (nine kinds of diagrams) including reverse engineering of methods into sequence diagrams. Plus HTML and RTF documentation—invoke it as part of your daily build, deliver always-up-to-date doc to your boss

Together/J

Download TJ Whiteboard from www.togetherj.com/jr/

"Having seen Together/J's direct use of Java code to support the models, I am forced to wonder why anyone would like to work in any other way! I do think that the strength of the product lies in its interoperability — it's a 'complete solution', not just a modeling tool." — Jon Collins, Senior Analyst, **Bloor Research**

"I really like Together/J. No other vender has even come close to making such an intuitive and convenient diagramming interface." —David Marine, Sr. Web Applications Developer, **NetSolve**

"We discovered in our evaluation that other modeling tools only get used during the front-end of a project. Yet we found Together/J to be the one UML modeler that our developers will use from start to finish. Why? It's the only one that delivers simultaneous (rather than batch-mode) round-trip engineering." — Curtis Chambers, **The Home Depot**

"I'm sold... I just finished doing the entire Use Case diagram for the project, (in the format I wanted it), publishing it to a web document on our web site, am working on the sequence and collaboration diagrams right now; and am scheduling a net video discussion group of all our users to discuss the use case designs that we posted... All with Together/J. Your product does verything we need to do now and more..." — Steve McConnell, **The Shaw Group**

"Together/J is an excellent modeling CASE tool for Java developers... It also meets the needs of real-world object modelers. Its support for language-specific features such as inner classes and JavaBeans, puts it miles ahead of other CASE tools that merely support the Java-language syntax... Together/J 3's support for key UML 1.3 diagrams, patterns, and Java-specific language features makes it a valuable addition to your Java development toolset." — **Jolt Award '99**

Together Software Corporation
1720 Leigh Drive
Raleigh, NC 27603
(919) 772-9350
www.togethersoft.com

rated jars top 1%

Editors' Choice JavaWorld 1998

Software Development Product Excellence Award

©1999 Together Software Corporation. All rights reserved.
Together is a registered trademark of Together Software Corporation
Java and EJB are trademarks of Sun Microsystems, Inc.

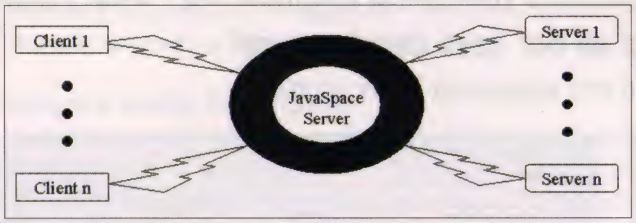


Figure 1. ***Java space overview.***

According to the requirements, AccountMaintenanceComponent and SecurityTradeComponent are used more than AccountTransferComponent. If memory is scarce on the deployment machine (Network Computer or other low-memory device), the developer may choose to buffer only AccountMaintenanceComponent and SecurityTradeComponent—leaving the AccountTransferComponent instance to load, construct, and initialize as needed. Buffering the AccountMaintenanceComponent on application startup improves the effective performance of the application, but increases the initial application start time (processor utilization) and requires additional memory.

If the deployment machine is upgraded with additional memory, the developer can opt to buffer additional components such as the AccountTransferComponent. The tradeoff is, again, additional application start time (processor utilization) and memory.

When to Buffer?

Once the architect has determined the appropriate instances to buffer, he or she must decide when to buffer the chosen instances during runtime.

There are two categories of instance buffering: synchronous and asynchronous. Synchronous buffering blocks program execution until the instance is completely buffered. This type of buffering creates instances immediately (usually more frequently used instances). Asynchronous buffering does not block program execution and buffers less frequently used instances in the background.

The implementation of instance buffering described here allows the developer to easily switch buffer strategies and even disable buffering altogether. This is very useful during development when the developer is testing specific instances, debugging code, and frequently restarting the application. See the "Implementation" section that follows for more details.

The following four buffer strategies are discussed:

- PayUpFront
- PayAsNeeded
- PayWhenConvenient
- PayOnLoan

Where "Pay" implies resource utilization.

PayUpFront (Synchronous)
Buffer immediately

Strategy Usage:
Deployment Environment
Fat Intranet PC Client
Resources
Network: Fast
Memory: High
Processor: Fast

Because resources are plentiful, the developer chooses to load, instantiate, and initialize all instances *up front*—on application startup, for instance. This synchronous buffer strategy increases the start time of the application, requires the most resource utilization at startup, but renders the most responsive application on buffer completion.

PayAsNeeded (Synchronous)
No buffering

Strategy Usage:
Deployment Environment
Thin Intranet NC Client
Resources
Network: Fast
Memory: Low
Processor: Slow

When memory and processor resources are scarce, the developer will not improve performance with instance buffering. Instead, the developer will choose to load, instantiate, and initialize each component as needed. This buffer strategy requires the least amount of resources on application startup. Although the application starts more quickly, a noticeable pause occurs every time the user accesses an unbuffered instance.

The implementation of instance buffering described here eliminates the need for code changes when switching to this synchronous buffer strategy—during development and debugging, for instance.

PayWhenConvenient (Asynchronous In-Process)
Buffer in the background with threads

Strategy Usage:
Deployment Environment
Fat Internet PC Client
Resources
Network: Slow
Memory: High
Processor: Fast

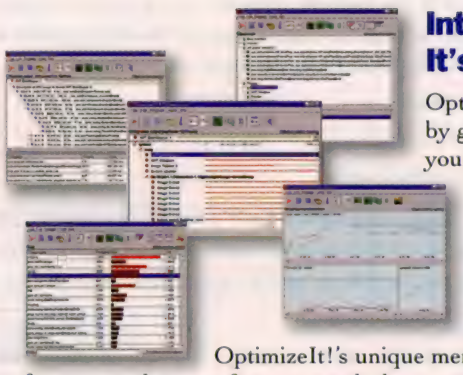
Because the memory and processor resources are favorable, the developer chooses to buffer some components in the background by spawning low-priority threads when convenient. This asynchronous strategy improves application start time, because application execution is not blocked.

Assuming most users choose to interact with the AccountMaintenanceComponent first, the developer chooses to fully load, instantiate, and initialize the AccountMaintenanceComponent on application startup. While the user is interacting with the low-interactive AccountMaintenance Component, low-level background threads load, instantiate, and initialize both the SecurityTradeComponent and AccountTransferComponent. Note: The AccountMaintenanceComponent thread is configured to execute at the highest priority.

This is the most efficient in-process buffer strategy. The user views the application as responsive—application start time is acceptable, and processor utilization is most efficient. Because the AccountMaintenanceComponent executes in the highest priority thread, the application remains responsive while other instances are buffered at a lower priority.

Advanced Java™ profiling with Optimizeit™!

Because you have better things to do with your time than track Java performance problems.



Introducing Optimizeit! 3.0 Professional.
It's Java Profiling Like You've Never Seen.

OptimizeIt! 3.0 Professional makes your Java profiling process easier by giving you the most comprehensive and accurate information on how your Java programs use CPU and memory. What you get is clear, real-time data and easy-to-read graphs allowing you to rapidly pinpoint performance problems such as bottlenecks, excessive object allocations or memory leaks.

OptimizeIt! is the only tool to include a CPU profiler allowing you to tune CPU time measurement to the most efficient level of detail for each test session – while maintaining a low overhead. OptimizeIt!'s unique memory profiler also provides unequaled features such as a reference graph that automatically highlights the references actually causing memory leaks.

But don't take our word for it. Check us out at www.optimizeit.com to download your free trial today, and experience how OptimizeIt! will help you improve the overall performance of your Java programs. And just maybe get you out of the office on time.

www.optimizeit.com

Optimizeit 3.0 Professional supports JDK 1.1.x and Java 2

©1999 Intuitive Systems, Inc. All rights reserved. Optimizeit is a registered trademark of Intuitive Systems, Inc. All registred trademarks belong to their respective manufacturers.

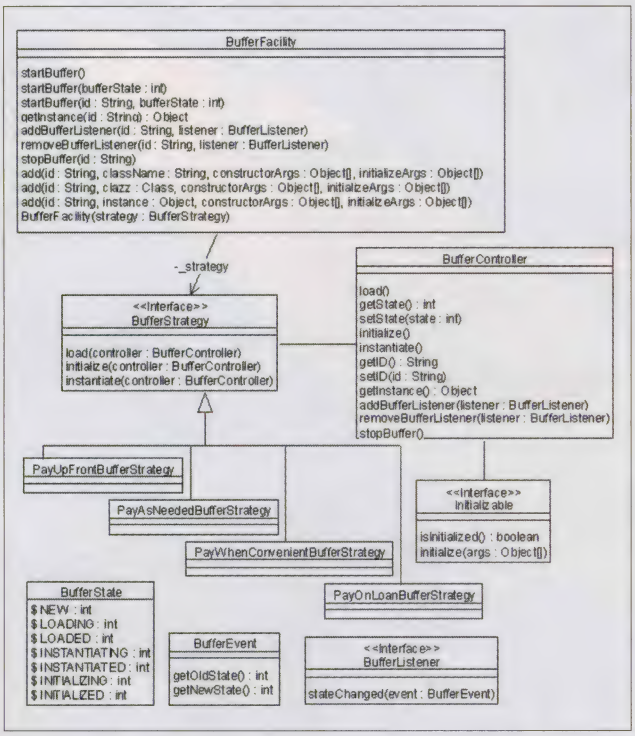


Figure 2. ***UML static class diagram.***

The implementation of instance buffering described here allows the main thread to receive a notification event when the background buffering completes. This allows the GUI developer to keep the SecurityTradeComponent and AccountTransferComponent tabs disabled until they initialize completely. The GUI developer can also choose to keep the tabs enabled. In this case, when the user selects an instance that is not initialized, the implementation will adapt to return the desired initialized instance as quickly as possible. See the "Implementation" section for more details.

PayOnLoan (Asynchronous Out-of-Process)
Offload buffering to other machines via JavaSpaces

Deployment Environment
Intranet NC Client
Resources
Network: Fast
Client
Memory: Low
Processor: Medium
Servers
Memory: High
Processor: Fast

Similar to the PayWhenConvenient strategy, PayOnLoan is an asynchronous strategy and notifies buffer listeners when buffering is complete. Because the memory and processor resources are scarce, however, the developer chooses to submit instances to other machines for buffering.

JavaSpaces is an appropriate mobile-object framework for this strategy. JavaSpaces is an entry-based, shared, concurrent server-processing model that allows many machines to appear as one large virtual supercomputer with virtually unlimited resources (see Figure 1). Understanding that a user usually interacts with the AccountMaintenanceComponent first, the developer chooses to fully load, instantiate, and initialize the AccountMaintenanceComponent on application startup.

While the user is interacting with the AccountMaintenanceComponent, instances are submitted to a JavaSpace server. A virtually unlimited number of machines can pull from JavaSpace to perform the initialization of SecurityTradeComponent and AccountTransferComponent. Once the worker machine completes the initialization task, the instance is returned to the JavaSpace server where it is returned to the client.

Off-loading these tasks to other machines allows the AccountMaintenanceComponent to service the user with minimal resource competition. This strategy offers tremendous scalability. Essentially, "distributed threads" are spawned across a federation of networked machines connected to a central JavaSpace UI Server.

Off-loading just the class loading or instantiating offers no improvement in performance. Classes need to be loaded and instantiated in every VM where the instance is manipulated—including the client and the worker server. Additional processor utilization is also required to serialize the mobile objects and initialization arguments around the network.

Although this strategy requires additional overhead, it allows many instances to be initialized in parallel on multiple machines. This technique is best suited for instances with resource-intensive initialization methods where the network overhead, additional class loading, and extra memory allocation are minimal compared to initialization.

Because network overhead is much larger when spawning these distributed threads, this strategy requires a very fast network connection. This strategy is not suited for a 56K dial-up connection deployment environment. Ideally, the client, the JavaSpace server, and the federation of worker servers require at least a LAN, T1, or other high-speed network connection.

To What Level?

The deeper the buffering, the better the performance; instance initialization is the deepest. The following levels of buffering are described within the limitations of the deployment environment and application-functional partitioning:

- Class loading
- Instance construction
- Instance initialization

Remote class loading requires a relatively small amount of memory, low processor utilization, and a large amount

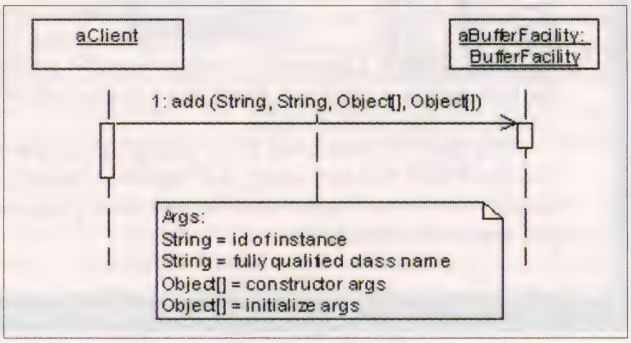


Figure 3. ***Adding a class to BufferFacility.***

Our Guts. Your Glory.

"GemStone/J will become the centerpiece of our advanced money transfer product. GemStone is clearly the application server leader."

— Richard Leurig, VP and CIO, MoneyGram and Travelers Express

NEW
White Paper on Internet Commerce by Industry Analyst John Rymer!

You're open for business on the World Wide Web. Sophisticated, custom-built Internet commerce applications are driving your business into greater growth and profitability. Life's looking good at the top. ▼ But are these applications robust and secure? They are, if you build them on the GemStone/J application server. It has the most comprehensive security solution available and the guts to really power your applications. World-class stuff built on the Java 2 security standards as well as integrated EJB and CORBA-based plumbing. Just ask our customers. ▼ Check out what a leading industry analyst, John Rymer of Upstream Consulting, thinks in his white paper **"Charting the Course of Internet Commerce Architectures."** Call 800-243-9369 or get it at **www.gemstone.com/guts**. Your glory may depend on it.

GEMSTONE®
So you can serve the world

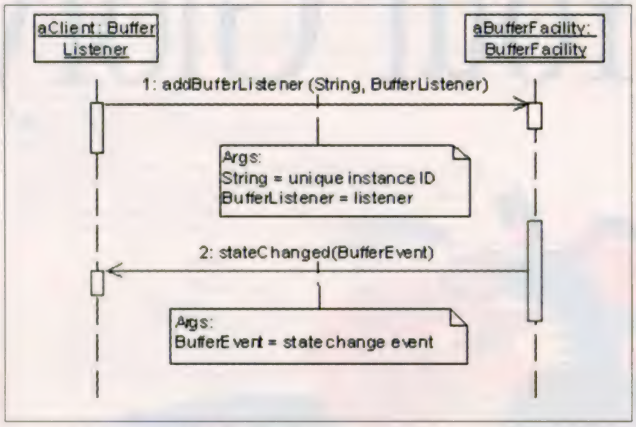


Figure 4. ***Listening for BufferEvents.***

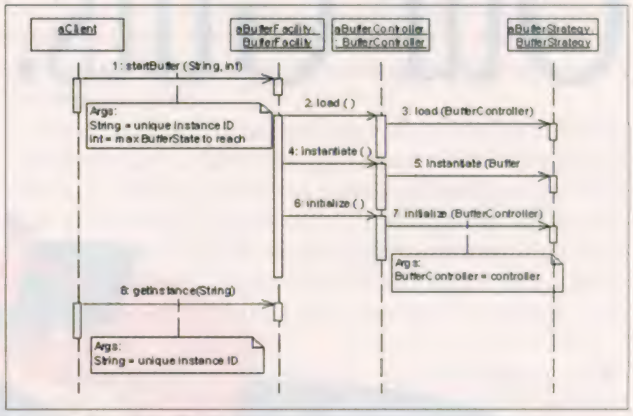


Figure 5. ***Start Buffering.***

of network traffic. Instance construction requires additional memory and processor utilization. Instance initialization requires a relatively large amount of memory and a large amount of processor utilization. The availability of each of these resources needs to be considered when choosing the level of buffering.

Again, resource-intensive initialization code should exist separately from the constructor of an instance. This allows an instance to be constructed without taking the hit of complete initialization. Separating these levels improves the flexibility of buffering. This fine-grained control allows the developer to adapt to any deployment environment.

Strategy Usage:
Deployment Environment
Thin Internet NC Client
Resources
Network: Slow
Memory: Low
Processor: Slow

Class Loading

Because resources are unfavorable—including network speed—the developer chooses to buffer only the class loading for AccountMaintenanceComponent, SecurityTradeComponent, and AccountTransferComponent. Deeper buffering would require

You Need It When?!

Because you need to quickly develop and deploy applications to the web, you need Apptivity™ from Progress® Software.

Apptivity is the industry's most complete development and deployment environment; it quickly generates Java-based applications that run anywhere.

As an authorized Apptivity™ Training Partner and total solution provider, JHL with its skilled trainers and developers will get your staff up to speed... your site up and running.

JHL ... from acquisition to deployment. **Let us be your total solution provider.**

JHL TECHNOLOGIES
Authorized Apptivity Training Partner
(Formerly JHL Computer Consultants)

Voice: (954) 845-9967
Fax: (954) 845-9661
jhlcomp@gate.net
www.jhlcomp.com

Training • Software Development • Mentoring

additional memory and processor utilization. Buffering dynamically loaded classes across a slow network will improve performance, assuming classes are individually downloaded and not part of an archive file such as a .JAR or .ZIP.

Instance Construction

Sometimes an application design requires more than class-load buffering. The application may need just references—even to uninitialized objects—to properly start up. These instances could be used to wire together event producers and consumers on startup, for instance, without requiring additional memory and processor utilization.

Instance Initialization

Initializing an instance requires the most memory and the most processor utilization. If application start time and available resources permit, a developer should buffer as many instances as possible to this level.

The implementation of instance buffering described here allows the developer to specify the deepest level to buffer for each instance. See the "Implementation" section below for more details.

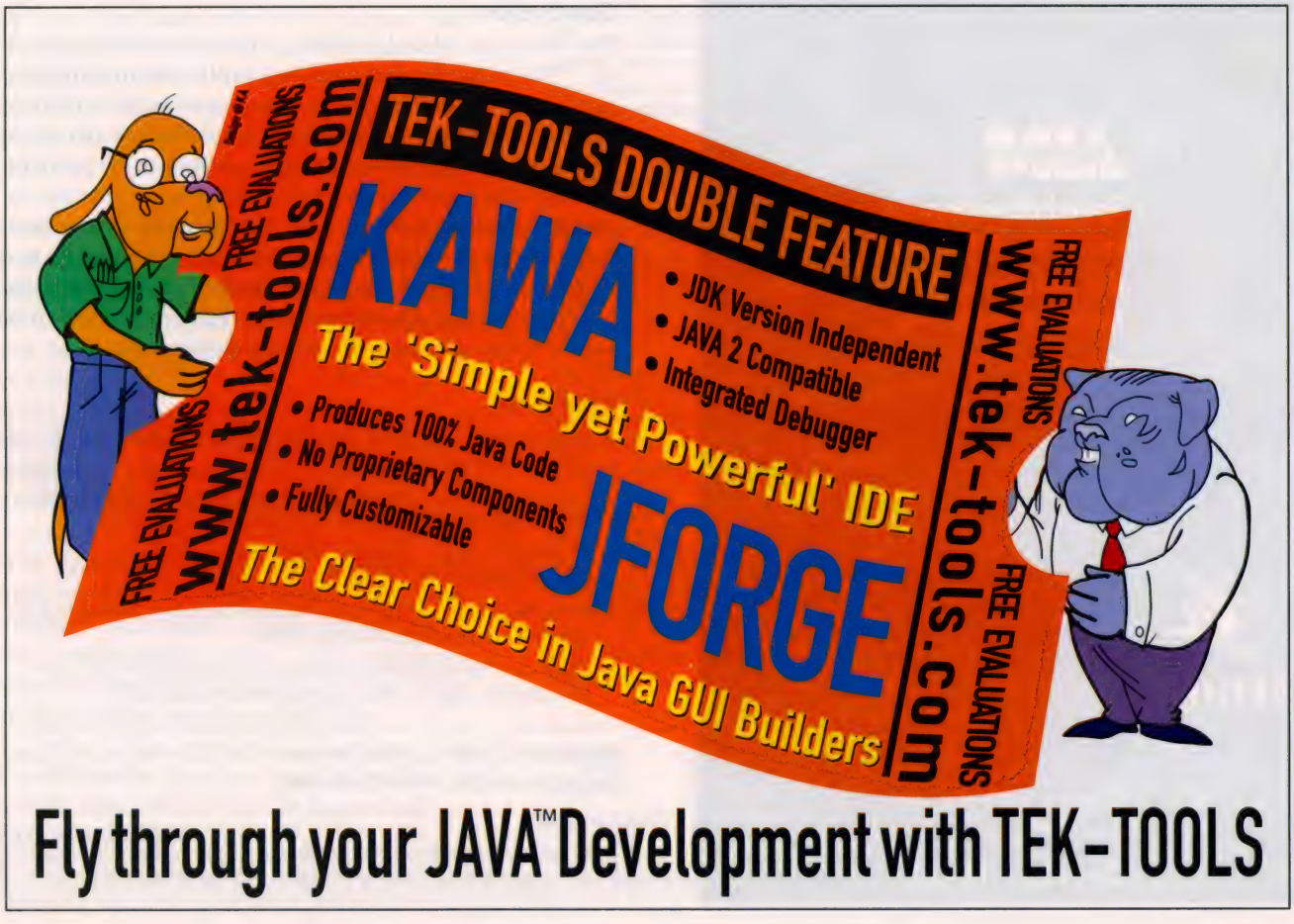
IMPLEMENTATION

Class Descriptions

- **BufferFacility:** Central façade for buffering.[2] This is the main instance used to add classes for buffering, add/remove BufferListeners, start buffering, and retrieve the initialized instance.
- **BufferStrategy:** Interface for all buffer strategies.[2] There exists a 1:1 relationship between a BufferFacility and its BufferStrategy.
- **BufferController:** Wraps each instance for buffering. Used by the buffer strategy to load, instantiate, and initialize the given class. There exists a 1:1 relationship between a BufferController and its instance.
- **BufferListener:** Listens for buffer state changes—Observer design pattern.
- **BufferEvent:** Event fired when the BufferController buffers the instance to a new state.
- **BufferState:** All possible buffer states.
- **Initializable:** Interface for all instances participating in initialization buffering. All buffered classes must implement this interface.
- **PayUpFrontBufferStrategy:** Implements Pay UpFront strategy as described.
- **PayAsNeededBufferStrategy:** Implements Pay AsNeeded strategy as described.
- **PayWhenConvenientBufferStrategy:** Implements PayWhenConvenient strategy as described.
- **PayOnLoanBufferStrategy:** Implements PayOnLoan strategy described above.

Design Goals

- Ability to start buffering with a class name (String), a Class, or an Object (uninitialized instance—see Figure 3).



Introducing a

wireless data solution

so fast,

you may need this.

AIM

Speed development and lower costs by taking AIM, the Enterprise Wireless Data Solution Software.

Buckle up and strap yourself in. If it's your job to deploy a wireless data solution, chances are you're not prepared for how quickly AIM Wirelessware by Aether can make it happen. With its built-in tools and easy-to-use SDKs, AIM speeds the integration of mobile devices with the enterprise. As the saying goes: Speed thrills. Take AIM. Call us today at (410) 654-6400 x255, or visit us at www.aethertech.com/AIM.

AETHER
REAL INTELLIGENCE. REAL TIME.

Come see us at PalmSource™ '99, October 19-22

Flexible Strategies for Optimizing

- Synchronous and asynchronous buffering.
- Similar interface for synchronous and asynchronous buffering.
- Notification of buffer state changes (required for a proper implementation of asynchronous buffering—see Figure 4).
- Ability to disable buffering during development.
- Ability to raise priority for specific instances.
- Flexibility through any combination of buffer type and level.
- Threadsafe.

Assumptions

- All buffered classes must implement Initializable.
- PayOnLoanBufferStrategy JavaSpaces implementation requires that all buffered classes implement Serializable.
- getInstance() will synchronously return a fully initialized instance regardless of strategy type (see Figure 5).
- All arguments (including initialization arguments) must be known up front when the class is flagged for buffering.
- Strings are used to uniquely identify an instance within the BufferFacility implementation.
- Because there is a 1:1 relationship between BufferFacility and BufferStrategy, multiple BufferFacility instances may coexist if different strategies within the same process are required.
- Disabling buffering from an asynchronous buffer strategy (for debugging purposes) is difficult, as the design relies on buffer-completion notification. While a BufferListener is not required for synchronous buffer strategies, it is required for asynchronous buffer strategies. Therefore, a BufferListener is recommended.

Conclusion

The faster line of code is the one that has already executed.[3] This is the fundamental principle behind buffering. Just as double-buffering and MediaTracker ready a graphic before it is needed, instance buffering readies any instance before it is needed. Instance buffering is a simple, straightforward, and powerful abstraction designed to improve application performance given any deployment environment. BufferFacility is a flexible, adaptive, and dynamic implementation of instance buffering for Java. The abstraction, however, is language independent. BufferFacility is part of ISA Services' Athens Architectural Components. ■

References

1. El-Rewini, H. and T. G. Lewis, *Distributed and Parallel Computing*, Manning, 1998.
2. Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
3. Loosley, C. and F. Douglas, *High-Performance Client/Server*, John Wiley & Sons, 1998.
4. Pree, W., *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995.
5. Oaks, S., "How can we use memory responsibly in Java?" *Java Report*, Vol. 3, No. 4, Apr. 1998, 96,94–95.

Christopher A. Fregly is a Senior Programmer for ISA Services, Inc. in Chicago. He can be contacted at chris@fregly.com.

Quantity reprints of this article can be purchased by phone: 717-399-1900 ext. 139 or by email: sales@rmsreprints.com.

MORE ABOUT PREVENTING ERRORS

by Adam Kolawa

Last month we presented a column about error prevention. The software industry is finally beginning to wake up to this promising new trend. Due to the enthusiastic response we received, we shall continue to present the various methods of error prevention that are available to developers.

The whole concept of error prevention is based on the idea that developers should protect themselves against writing dangerous code and should avoid errors before they happen. One of the most effective ways for developers to prevent errors is to implement coding standards.

Some developers may need to familiarize themselves with coding standards if they have not used them before. Coding standards are language-specific rules that, if followed, reduce the opportunity for developers to introduce errors into code. Developers who follow coding standards have an easier time avoiding sequences or constructs that are dangerous.

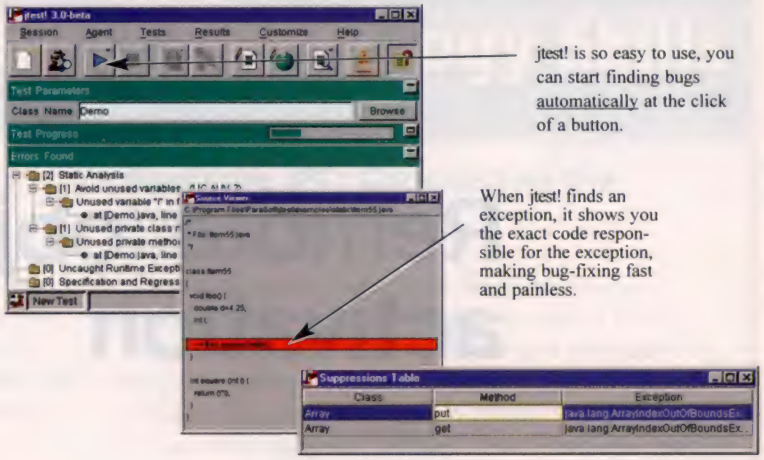
Certain constructs are dangerous because they are error-prone. Perhaps the most dangerous aspect of these constructs is how harmless they seem at a glance. You may examine a segment of code that contains dangerous constructs and not notice any faults. However, when you put that code in the context of a larger program, it may enter interactions that lead to errors.

Therefore, it is essential that developers avoid dangerous constructs by following coding standards. Exercising a bit of caution early in a project can save developers from hours of tedious, wasteful repair work later on.

Over the months to come, we shall expose problematic constructs in Java code and provide you with ways to avoid them.

We at ParaSoft look forward to your continued readership. I am certain that these principles will make a real difference in your development cycle. For archived copies of previous columns, please visit our Web site at www.parasoft.com.

Adam Kolawa is President and CEO of the ParaSoft corporation. He can be reached at ukola@parasoft.com



jtest! is so easy to use, you can start finding bugs automatically at the click of a button.

When jtest! finds an exception, it shows you the exact code responsible for the exception, making bug-fixing fast and painless.

With jtest!, it's easy to focus on the exceptions that are most important to you. You can customize reports to suppress exceptions by type, name, and enclosing method.

Would you believe that jtest!™ can take your brand-new class and test it on its own, automatically?

*"jtest!™ represents a dramatic step forward in Java™ testing. As developers move toward serious application development, it is absolutely critical for them to incorporate tools such as jtest! in their development cycle." -- **Will Iverson Symantec***

jtest!™ is the world's first automatic class testing tool for Java™. Testing on its own with one click from the developer, jtest! automatically passes unexpected inputs to each class and validates that the inputs will not cause the program to crash. Using patented Test Generation System Technology (#5,784,533 & 5,761,408), jtest! tests all methods of a class. When jtest! finds an input that generates an uncaught runtime exception, it reports to the developer the arguments that cause the problem, the stack trace for the exception found, and the calling sequence that shows the exception happening. The testing process with jtest! prevents errors that could cause problems when classes are integrated into modules.

For version 3.0, ParaSoft has added three impressive new testing capabilities to jtest!: black-box testing, regression testing, and static analysis.

Black-box testing, which ensures that each class behaves according to specification, is now a fully automatic process with jtest!. During black-box testing, jtest! automatically provides a set of inputs based on sophisticated analysis and executes the class with the inputs. The user can also provide a set of inputs that will be tested along with the automatically generated inputs. jtest! will then report the outcomes, which the user can verify with one click of the mouse.

jtest! automates regression testing, a process similar to black-box testing. During regression testing, jtest! helps make sure that developers do not re-introduce errors to their code. jtest! remembers outcomes from previous runs and compares outcomes every time the class is tested, reporting an error if an outcome changes. Developers can use jtest! to make sure they are not letting their code regress into previous errors when they make changes.

In addition, you can now enforce Java coding standards automatically with the new static analysis feature of jtest!. jtest! parses the Java source of a class and applies to it a set of rules. Any violations to these rules are reported as errors. Enforcing coding standards at every stage of the development process will lessen the amount of debugging developers must perform later in the development cycle.

If you would like the satisfaction of using jtest!, visit www.parasoft.com/errors.htm or call ParaSoft toll-free at (888)305-0041 for more information.

ParaSoft®

www.parasoft.com/errors.htm

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. jtest! is a trademark of the ParaSoft Corporation.

Remote object activation

Govind Seshadri
gseshadri@str.com

FEW DENY THAT, today, RMI is among the most useful and versatile technologies within the Java pantheon. Arguably, this has not always been the case, as there were some glaring lacunae within the RMI architecture implemented under earlier versions of Java. Under JDK 1.1, every RMI server had to be started manually prior to its usage by remote clients. Also, if a remote object crashed during operation, a client could not really resume "where it left off," as the previous state of the remote object was unrecoverable. But all that is history. I will examine the details behind a significant enhancement to the RMI architecture under the Java 2 platform that helps overcome these limitations.

Understanding Remote Object Activation

While it sounds simple, starting remote servers prior to their usage may not always be a practical proposition. For instance, it is highly inefficient to have hundreds of resource-intensive remote objects running in memory all the time—especially if they are used only on an occasional basis. What's worse, they may even turn out to be a logistical nightmare if your enterprise's distributed computing environment demands hundreds or thousands of remote objects. As we can see quite clearly, the old approach of prestarting an RMI server becomes unmanageable rather quickly in a real-life scenario.

Luckily for us, however, with the advent of Java 2, the headaches associated with developing fairly large-scale distributed systems using RMI have now been greatly alleviated. Specifically, with the help of the new Remote Object Activation (ROA) feature, you no longer have to worry about starting up remote objects prior to their invocation.

By extending java.rmi.activation.Activatable and making use of the new RMI activation daemon rmid, RMI developers can now register information about remote object implementations that should be instantiated only when necessary, rather than running all the time. The ROA mechanism accomplishes this by what is known as *lazy activation*. Lazy activation implies that the passive remote object is activated by rmid, by loading its constituent classes into a JVM only on the receipt of the client's first remote method invocation, and not until then. rmid—which runs on the same host as the activatable object and the RMI name server, rmiregistry—also needs to be configured with information that will allow it to locate and load the classes used by the remote object. This task is usually delegated to a separate setup or bootstrap program, after first starting up rmid and rmiregistry. Activatable objects can optionally be associated with a persistent state. This can be provided to the remote object during activation to reestablish its state information. The good news is, choosing to make your remote objects is strictly a server-side decision; RMI clients don't care whether the remote objects are activatable or not.

Developing an Activatable Remote Object

ROA can be a fairly complex topic for the novice RMI developer. If you are not familiar with Java remote objects, it may be useful for you to read my earlier article on distributed Java and RMI.[1] Assuming you are familiar with the basics of RMI, let us take a closer look at activatable remote objects. The best way to understand ROA is to see it in action via a concrete example. As with any RMI object, the activatable object in our example implements a remote interface. Here, CountInterface (shown in Listing 1), contains a single remote method getCount() and basically allows a client to obtain a count of the number of times the remote object has been accessed.

The class for our activatable server object is shown in Listing 2. CountServer not only maintains a running count of the number of accesses

Govind Seshadri is a Senior Technical Architect with STR in Milwaukee. He is also the author of *Enterprise Java Computing*.

Advance Your Object Technology Skills

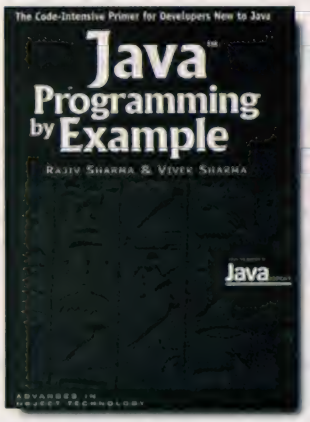
Java for the COBOL Programmer

E. Reed Doke** and **Bill C. Hardgrave

Java for the COBOL Programmer describes the features and syntax of Java while contrasting them with their parallels in COBOL. With each example, the authors present a COBOL program that performs a certain function and develop a Java program that accomplishes the same task. Sample programs are available on an accompanying CD-ROM.

Advances in Object Technology 20

1999 350 pp. 0-521-65892-6 Paperback $44.95



Java Programming by Example

Rajiv Sharma** and **Vivek Sharma

Introduces software developers to Java, using programming demonstrations and wry humor. The authors provide the information developers need to quickly create applets, applications, multi-file programs, animation and GUIs.

Advances in Object Technology 17

1998 446 pp. 0-521-64442-9 Paperback $34.95

Enterprise Java Computing

Applications and Architecture

Govind K. Seshadri

An ideal hands-on reference written by a seasoned Java expert. Chapters deal with practical design and deployment issues, such as Relational Database Integration, Server-side Java, Distributed Object Computing, and Real-World Deployment.

Managing Object Technology Series 22

1999 360 pp. 0-521-65712-1 Paperback $39.95

The Elements of Java Style

Alan Vermeulen, et al.

Just as Strunk and White's *The Elements of Style* provides rules of usage for the English language, this text furnishes a set of rules for Java practitioners. While illustrating these rules with parallel examples of correct and incorrect usage, the authors offer a collection of standards, conventions, and guidelines for writing solid Java code that will be easy to understand, maintain, and enhance.

Advances in Object Technology 26

1999 125 pp. 0-521-77768-2 Paperback $24.95

Hardcore JFC: Conquering the Swing Architecture

Mitch Goldstein

A comprehensive guide to the functionality and practical use of the Java Foundations Classes (JFC), a component library designed for the Java programming language and operating environment. The text features concrete illustrations of how to enhance the JFC architecture: leveraging the "Model/View/Controller" paradigm, customizing and developing new components and techniques for creating "look and feel" user interface classes. Goldstein introduces explanations of design and development techniques with effective, well-constructed, and reusable example code.

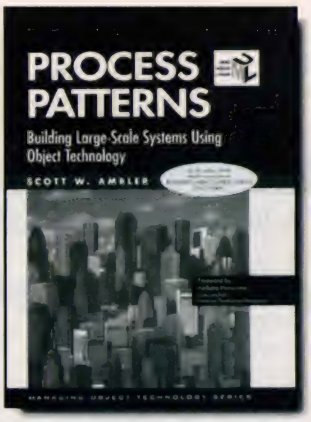
Advances in Object Technology 23

1999 600 pp. 0-521-66489-6 Paperback $44.95

Process Patterns

Building Large-Scale Systems Using Object Technology

Scott W. Ambler



Written by one of the best known object-oriented practitioners, *Process Patterns* shows how to successfully deliver large-scale applications using object technology, and describes how one develops applications that are easy to maintain and to enhance. Geared toward developers and project managers in medium to large-size organizations.

Managing Object Technology Series 15

1998 582 pp. 0-521-64568-9 Hardback $44.95

More Process Patterns

Delivering Large-Scale Systems Using Object Technology

Scott W. Ambler

Presents process patterns for the second half of the development life cycle of large-scale, object-oriented system development: the Deliver phase and the Maintain and Support phases. Covers major management issues, such as people and risk management, and quality assurance.

Managing Object Technology Series 19

1999 313 pp. 0-521-65262-6 Hardback $44.95

Available in bookstores or from

CAMBRIDGE UNIVERSITY PRESS | SIGS BOOKS

40 West 20th Street, New York, NY 10011-4211 Call toll-free 800-872-7423 Web site: http://www.cup.org
MasterCard/VISA accepted. Prices subject to change.

LISTING 1.

Remote interface for activatable remote object.

```
package com.jr.active;

import java.rmi.*;

public interface CountInterface extends Remote {

   public Integer getCount() throws RemoteException;
}
```

by RMI clients, but is also activatable—i.e., its instantiation is deferred until the first remote method invocation. Further, it also makes use of MarshalledObject to regain its state—even if it is restarted after a server shutdown.

Now, let us dissect the activatable remote server a little and see how it differs from a conventional RMI server. Consider the line:

```
public class CountServer extends Activatable

 implements CountInterface
```

Unlike regular remote objects that subclass java.rmi.server.UnicastRemoteObject, the class for an activatable object *usually* needs to extend java.rmi.activation.Activatable. (I emphasize usually, because you can also create an activatable server object by exporting a remote object using the static method Activatable.exportObject().)

Observe the constructor for CountServer:

```
public CountServer(
  ActivationID id, MarshalledObject data)
    throws RemoteException,
    ClassNotFoundException, IOException {
...
}
```

The constructor is automatically invoked by the RMI activation daemon rmid whenever a new instance of this remote object needs to be created, and really serves two purposes. It first registers the remote object with the activation system, and then exports it on an anonymous port. The ActivationID passed in by rmid represents a unique identifier for the remote object to be activated and includes a remote reference to the object's activator.

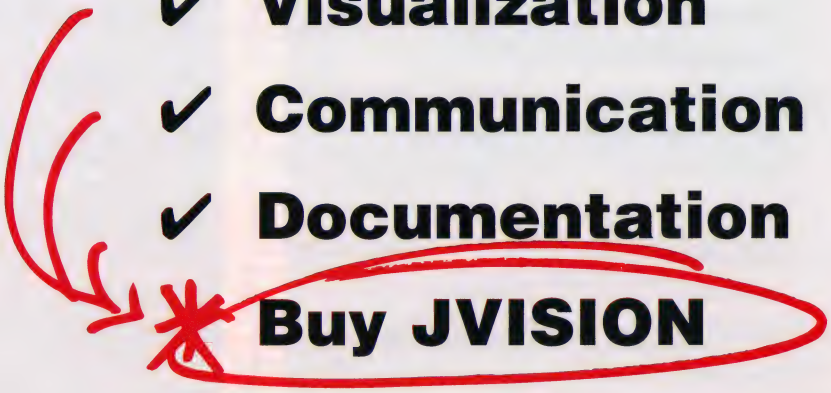
The MarshalledObject parameter here can also optionally contain any serialized object. In this example, we use it to supply the name of the file containing some previously persisted state information. The great thing about using the MarshalledObject is that now, your remote objects can

LISTING 2.

Activatable remote object.

```
package com.jr.active;

import java.rmi.*;
import java.rmi.activation.*;
import java.io.*;

public class CountServer extends Activatable
   implements CountInterface {

   private Integer count;
   private File storage;

   public CountServer(ActivationID id, MarshalledObject data)
  throws RemoteException, ClassNotFoundException, IOException {

  super(id, 0);

     // The MarshalledObject is used to pass any previously
     // persisted state information
  storage = (File) data.get();

  if (storage.exists()) {
     // If MarshalledObject is a serialized file, then use the
     // state info within that to initialize the object

       this.restoreState();
  } else {
     // Remote object is starting for the first time
     count = new Integer(0);
  }
   }

   public Integer getCount() throws RemoteException {
    this.saveState(); //increase and persist counter object
    return count;
     }


private void restoreState() throws IOException,
  ClassNotFoundException {

  File f = storage;
  try {
    FileInputStream fis = new FileInputStream(f);
      ObjectInputStream ois = new ObjectInputStream(fis);
      count = (Integer)ois.readObject();
      ois.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}

   private synchronized void saveState() {

       FileOutputStream fos = null;
       ObjectOutputStream oos = null;

      File f  = storage;
      try {
       fos = new FileOutputStream(f);
       oos = new ObjectOutputStream(fos);
       count = new Integer(count.intValue()+1);
       oos.writeObject(count);
       oos.close();
      } catch (IOException e) {
       e.printStackTrace();
       }
   }
}
```

Programming complex projects requires:



- ✔ Visualization
- ✔ Communication
- ✔ Documentation
- Buy JVISION

JVISION™

$99 or 30 day try before you buy

OTJ™

Object Technology for Java™

Free Download

Visualize your architecture and document projects effortlessly with instant UML diagrams from Java™

Visual Café™ integration add-on

Documentor™ add-on product allows HTML export and instant website creation for multiple diagrams, plus JavaDoc™ integration

products@object-insight.com

Training:

Understanding Objects
Using UML with Java™
Designing Patterns for Java™
OO Software Development
Introduction to Java™
Java™ Basics
Using IBM VisualAge™ for Java™
Java™ and the Internet
Java™ Technology Forecast

training@object-insight.com

OBJECT™ insight

www.object-insight.com

Java™ and JavaDoc™ are trademarks of Sun Microsystems. VisualAge™ is a trademark of IBM. Visual Café™ is a trademark of Symantec. Object Insight™, JVISION™, Documentor™ and OTJ™ are trademarks of Object Insight, Inc.

ADVERTISER INDEX

Advertiser	Page	URL	e-mail
Ad Astra Engineering	33	www.JumpingBeans.com	crygaard@AdAstraEng.com
Aether Systems Inc.	58	www.aethersystems.com	
AvantSoft Inc.	68,79	www.avantsoft.com	jobs@avantsoft.com
BEA	2	www.weblogic.beasys.com	info@weblogic.com
C++ World	73	www.sigs.com	conferences@sigs.com
Cambridge University Press	61	www.cup.org	marketing@cup.org
Career Central	79	www.careercentral.com/dev	
Cerebellum	43	www.cerebellumsoft.com	info@cerebellumsoft.com
Computer Associates	13	www.cai.com/ads/jasmine/dev	CAinfo@CAI.com
Cyrus Intersoft	9	www.cyrusintersoft.com	leslie@syrusintersoft.com
Cysive	67	www.cysive.com	hr@cysive.com
DevelopMentor	49	www.develop.com	
Fiorano	66	www.fiorano.com	info@fiorano.com
Force 5	45	www.force5.com	steve@force5.com
Gemstone	55	www.gemstone.com/guts	info@gemstone.com
IAM	47	www.iam-training.com	java@iam-training.com
insignia	1	www.insignia.com	evm@insignia.com
InstallShield	CV 3	www.installshield.com	tom@installshield.com
Intel	19	http://developer.intel.com/vtune	mpl@mailbox.intel.com
Intuitive	53	www.optimizeit.com	optimizeit@intuisys.com
Java Developers Conference	69	www.javadevcon.com	conferences@sigs.com
JHL	56	www.jhlcomp.com	jhlcomp@gate.net
KL Group	CV 4, 35	www.klgroup.com	info@klgroup.com
Kovsky	21	www.softwaresummit.com	
Loox	15	www.loox.com	info@loox.com
Numega	23	www.numega.com	julie.corvo@numega.com
Object Design	31	www.objectdesign.com/javlin	info@objectdesign.com
Object Insight	63	www.object-insight.com	ron@object-insight.com
Object International	51	www.oi.com	coad@oi.com
Object People	40-41	www.objectpeople.com	rausten@objectpeople.com
Parasoft	59	www.parasoft.com	Janusz@parasoft.com
Persistence	5	www.persistence.com	info@persistence.com
Rogue Wave	39	www.roguewave.com	sales@roguewave.com
RSA	CV 2	www.rsa.com/win	lstrand@rsa.com
Sales Vision	37	www.salesvision.com	rhhecht@salesvision.com
Sanchez	78	www.sanchez.com	jobs@sanchez.com
Silver Stream	25	www.silverstream.com	info@silverstream.com
Software Development 99	75	www.sdexpo.com	sd99@mfi.com
Sybase	27	www.sybase.com/easerversuccess	sales@sybase.com
Tek-Tools	57	www.tek-tools.com	contact@tek-tools.com
Tidestone Technologies	28-29	www.tidestone.com	sales@tidestone.com
XML One	71	www.sigs.com	conferences@sigs.com
Zero G Software	7	www.zerog.com	info@zerog.com

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

BPA INTERNATIONAL

ADVERTISING/SALES

Senior Sales Director/Western Region Manag
Miles E. Silverman
• phone (212)242-7447 • fax (212)242-7574
• email msilverman@sigs.com

Advertising Sales Manager, Eastern Region
Brian Levine
• phone (212)242-7447 • fax (212)242-7574
• email blevine@sigs.com

Advertising Sales Manager
Development Solutions Buyer's Guide
Maurine Blake
• phone (212)242-7447 • fax (212)242-7574
• email mblake@sigs.com

CIRCULATION

Vice President Circulation
Christine Oldenbrook

Assistant Circulation Manager
Richard Bozza

FINANCE

Vice President Finance
Margherita R. Monck

Controller
David Chatterpaul

Accounts Receivable Credit Manager
Rey Shields

Accounts Payable
Bibi Budhram

MANUFACTURING

Vice President, Manufacturing
Brad Stauffer

Assistant Production Manager
Michele Costabile

FOUNDER
Richard P. Friedman

101 communications

SIGS PUBLICATIONS

WORLDWIDE OFFICES

Jane Major, SIGS Ltd., Brocus House,
Parkgate Road, Newdigate, Surrey RH5 5AH, UK
• phone 011.44.1.306.631.331 • fax 011.44.1.306.631.696
• email enquiries@sigs.com

Sabine Schiffman, SIGS Conferences GmbH, Odenthal
Str. 19, D-51465 Bergisch Gladbach, Germany
• phone 49.2202.9372.0 • fax 49.2202.9372.2
• email 100634,2070@compuserve.com

preserve state even between successive restarts. Observe that the constructor also checks whether the file passed via MarshalledObject exists, indicating the presence of a previously persisted state. If it does exist, the method restoreState() is invoked, causing the activatable object to initialize itself from the persisted data.

Because our remote object implements CountInterface, it defines getCount(), which increments a counter specifying the number of times the object was accessed. For each client invocation, this method updates and saves the count data to the file that was passed as the MarshalledObject, before returning the count to the client.

You may also have noticed that unlike a conventional RMI server, an activatable remote object does not contain a main() method. There is no need for one, as the server is now instantiated by rmid, instead of being manually started from the command line.

Setup Program

Before activatable objects can be instantiated, the rmid and rmiregistry services must be properly initialized with information regarding these objects. First, we need to initialize rmid with some URL-based information that will allow it to locate and load the remote object's constituent classes before instantiating them. Second, we have to register a remote reference (an instance of the generated stub class) for the activatable class within the rmiregistry, and associate it with an URL-based identifier. A setup program that is executed for each activatable object usually performs this configuration work. Listing 3 demonstrates the setup program for our activatable remote object.

Because our activatable object serializes its state information to a file, we have to explicitly grant this level of access, bypassing the default security sandbox. For testing, you can give *carte blanche* access to the setup program by disabling the security mechanism with the following policy file:

```
grant {
    permission java.security.AllPermission;
};
```

Once we have created the policy file, we need to initialize the java.security.policy property with its location. For example:

```
props.put("java.security.policy","/policy");
```

assumes that the policy file exists in the root directory (i.e., C:\ on win32 systems).

The activation system provides great flexibility when activating remote objects through the use of activation groups. Each activation group is associated with its own JVM. Thus, it is possible to run groups of activatable objects within different JVMs by associating them with a distinct ActivationGroupID:

```
ActivationGroupDesc.CommandEnvironment ace =
  null;

ActivationGroupID agi=
      ActivationGroup.getSystem().registerGroup(
                  new
ActivationGroupDesc(props, ace));
```

Consider the line:

```
MarshalledObject data=
  new MarshalledObject(
  new File("/Count.ser"));
```

LISTING 3.

Bootstrap program for activatable remote object.

```
package com.jr.active;

import java.rmi.*;
import java.rmi.activation.*;
import java.util.Properties;
import java.io.File;

public class Bootstrap {

  // This class registers information about the CountServer
  // class with rmid and the rmiregistry

  public static void main(String[] args) throws Exception {
CountInterface counter;

System.setSecurityManager(
  new RMISecurityManager());

Properties props = new Properties();

// Because of the Java 2 security model,
// file writing has to be
// explicitly allowed
props.put("java.security.policy", "policy");

    ActivationGroupDesc.CommandEnvironment ace = null;

ActivationGroupID agi=
  ActivationGroup.getSystem().registerGroup(
              new ActivationGroupDesc(props, ace));

// Indicate filename we want to persist as the
// Marshalled object
MarshalledObject data=
      new MarshalledObject(new File("/Count.ser"));
String location = "file:///some_path/";

ActivationDesc desc = new ActivationDesc(agi,
"com.jr.active.CountServer",
location, data);

counter = (CountInterface)Activatable.register(desc);

//bind within rmiregistry
Naming.rebind("/CountServer", counter);

System.exit(0);
  }
}
```

Here, MarshalledObject is indicated as being the file Count.ser, which is created in the root directory. This file is then used by our activatable remote object to persist the updated count data after each client invocation. Although we could have probably hard-coded the filename within the remote object itself, MarshalledObject provides a flexible mechanism for passing any kind of initialization object to the activatable remote object.

All the information required to activate a remote object is stored within an instance of the activation descriptor ActivationDesc. This activation descriptor is then interrogated by rmid to obtain the necessary information during object activation.

The line:

```
ActivationDesc desc = new ActivationDesc(
 agi, "cup.chap6.activation.CountServer",
 location, data);
```

creates an ActivationDesc object that stipulates the fully qualified class name for the activatable remote object, the location of its code, the object's group identifier, as well as any object-specific initialization data. The activatable object is then registered with rmid by passing the activation descriptor as a parameter to the static method Activatable.register() as:

```
counter =
 (CountInterface)Activatable.register(desc);
```

The register() invocation returns the Remote stub or client-side proxy for the activatable object. Basic RMI mechanics dictate that the client must obtain a handle to the remote object by performing a lookup on the rmiregistry. Thus, we bind the stub for the remote object within rmiregistry and permit its access via an URL-based name through:

```
Naming.rebind("/CountServer", counter);
```

The Client Applet

Listing 4 shows the client applet which is used to access the activatable remote object. Notice that there is nothing new happening here—the RMI client remains the same for a regular remote object as well as for an activatable one.

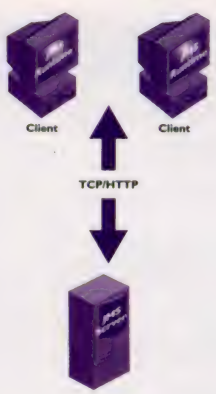
Demonstrating Remote Object Activation

Each time a client makes a remote method invocation, the activatable remote object updates its counter and stores the state in the form of serialized data to the file specified by the MarshalledObject. Now that you have the requisite example programs ready, you can compile them and generate the RMI stub and skeleton files as usual:

```
javac -d . *.java
rmic -d . com.jr.active.CountServer
```

JMS High Performance
Fiorano 3.0
Enterprise Messaging Server
Secure SCALABLE

Build secure, reliable and highly available distributed systems using standards-based Java APIs.



Communication, Security, Administration

- 100% Java implementation of **JMS** (Java Message Service) publish/subscribe and point-to-point communication APIs.
- Available C++ runtime library allows C++ and Java programs to communicate seamlessly.
- Lightweight, efficient and robust environment; thin runtime access to messaging services.
- ACL-based security based on Java 2 standards allows external control over system resources. Integrated SSL support for encrypted data transfers.
- Comprehensive Administration API for remote monitoring of servers.
- Integrates with popular Application Servers.

Guaranteed Delivery, Scalability and Availability

- Automatic store-and-forward ensures guaranteed message delivery across faulty networks.
- Server clustering allows support for thousands of concurrent client connections. Built-in support for load balancing, failover protection and high-availability allows clients to automatically connect to the least loaded server in a cluster.

Fiorano

Fiorano Software, Inc.
236 N. Santa Cruz Ave., Suite #213 • Los Gatos, CA 95030
Phone: (408) 354-3210 • Fax: (408) 354-0846
email: info@fiorano.com • www.fiorano.com

Before you can run the setup program, you need to start the RMI name server rmiregistry and the activation daemon rmid:

```
start rmid
start rmiregistry
```

Assuming that you had compiled the example source files in the path c:\examples and placed the security policy file in the root directory, you can execute the setup program as:

```
start java -Djava.security.policy=\policy
    -Djava.rmi.server.codebase=file:\examples\
      com.jr.active.BootStrap
```

The counter.html file shown in Listing 5 should work fine for the purpose of loading your applet.

The RMI applet can then be started as:

```
start appletviewer counter.html
```

Note that you do not have to first start up the remote server. The activatable object was already registered within the registry when we ran the setup program. If you keep executing CounterApplet repeatedly for a minute or two, you should eventually see the output shown in Figure 1.

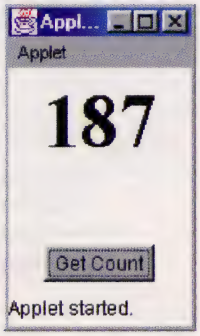


Figure 1. ***Output of repeated execution of CounterApplet.***

Now let us observe an interesting phenomenon. Without closing the appletviewer window, shut down rmiregistry and rmid. After that, restart both of them, and run the setup program as before. You should observe that the applet can resume where it left off—i.e., the RMI stub reference holds well even if the server goes down. Also, the activation system automatically restarts the remote server, while reading in the previous state from the serialized file passed via MarshalledObject.

Design Considerations

Remote object activation is an extremely powerful feature and should empower us to build much more flexible distributed systems. Although activatable and non-activatable remote objects can coexist in the same environment, they introduce a new level of complexity to the remote object lifecycle management issues. For instance, activatable remote objects can be deactivated via the method Activatable.unexportObject(). If the target remote object happens to be the last one present within its activation group, then one of the consequences of this deactivation call would result in the shutting down of the JVM! This could in turn lead to RMI clients that access

Hot Java

Cysive. We're premier builders of e-business architectures. Over 85% of our consultants are Java certified. Formerly known as Alta Software, our track record for quality and reliability is unmatched in the e-business industry. What sets us apart from the competition is our unparalleled talent and our incisive use of the latest tools and technologies to deliver working solutions every time.

Cysive is expanding rapidly and our biggest challenge is maintaining the caliber of our talent base as we grow. We seek highly skilled team players with demonstrated experience designing, developing and delivering technical solutions using **Java, CORBA, XML** and OO design methodologies. Positions are available in Reston, VA, Dallas, TX, Atlanta, GA, Chicago, IL and Northern California. Willingness to travel is required, but you'll find the technical challenges well worth it.

Cysive offers highly competitive compensation packages that include quarterly bonuses, stock options and exceptional training opportunities. Cysive is like no place you've worked before. For more information, please send your resume or letter of interest to:

E-mail: hr@cysive.com
Visit our Web site at: www.cysive.com

Cysive.™

net business. net growth. net results.

LISTING 4.

Client applet for the activatable remote object.

```
package com.jr.active;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.rmi.*;

public class CounterApplet extends Applet {

  public void init() {
      setLayout(new BorderLayout());
      setSize(114,138);

      hits = new java.awt.Label("",Label.CENTER);
      hits.setFont(new Font("Serif", Font.BOLD, 48));
      add("North",hits);

      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      countButton = new java.awt.Button();
      countButton.setLabel("Get Count");
      p.add(countButton);
      add("South",p);

      MyAction lMyAction = new MyAction();
      countButton.addActionListener(lMyAction);

      try {
          counter = (CountInterface)
        Naming.lookup("//"+
    getCodeBase().getHost()+"/CountServer"
    );
      } catch (Exception e) {
          e.printStackTrace();
      }
  }

  CountInterface counter=null;
  Label hits;
  Button countButton;

  class MyAction implements ActionListener  {
      public void actionPerformed(ActionEvent event)    {
          Object object = event.getSource();
          if (object == countButton)
              countButton_ActionPerformed(event);
      }
  }

  void countButton_ActionPerformed(ActionEvent event) {
      try {
          Integer result = (Integer)counter.getCount();
          hits.setText(result.toString());
      } catch (Exception e) {
          e.printStackTrace();
      }
  }
}.
```

LISTING 5.

Counter.html.

```
<html>
<body>
<applet code=com.jr.active.CounterApplet.class
  width=300 height=300>
</applet>
</body>
</html>
```

non-activatable remote objects left with dangling stub references.

Conclusion

Using the Activatable class and the activation daemon rmid, you can design remote object implementation that can be executed on-demand, rather than running all the time. This feature allows us to design highly scalable distributed systems while making efficient use of system resources. rmid provides a default JVM from which other JVM instances may be automatically spawned. We have also seen how the MarshalledObject can be used for passing initialization data through the activation descriptor, without having to hard-code these values within the remote object itself. ■

Reference

1. Seshadri, G., "Distributed Java Computing Using RMI," *Java Report*, Vol. 2, No. 10, Oct. 1997, pp. 39–46.

Quantity reprints of this article can be purchased by phone: 717-399-1900 ext. 139 or by email: sales@rmsreprints.com.

HOT COURSES
Enterprise JavaBeans (EJB)
XML for Java programmers
Who is the leader in applying Java to enterprises?
IBM
State-of-the-art Java curriculum
Core Java
JavaBeans
Swing/JFC
OOAD/Java
Java/CORBA
Java Security JDK 1.2
Enterprise Connectivity
Java JDK 1.2 new features
Who taught advanced Java courses to 2000 IBMers in the last 3 years?
AvantSoft, Inc.
An avant-garde training and consulting company
Our courses support new Certification Initiative for Enterprise Development
To schedule on-site Java classes,
Call: (408) 530-5705
Web: www.avantsoft.com
Avantsoft is the leader in exceptional quality Java training

Now in our 4th successful year!

PUTTING JAVA TO WORK!

October 17–21, 1999 Fairmont Hotel, San Jose, CA

SIGS Conference FOR JAVA™ Development

Serious About Building Reliable and Robust Java based Systems?

There's only one place to be this fall.

The place to learn and explore

Join thousands of your colleagues at one of the Java industry's best attended events – **SIGS Conference for Java™ Development** – now in its 4th successful year! With an all new technical program geared for both beginner and experienced Java users, **SIGS Conference for Java™ Development** will guide you to new levels of productivity- helping you infuse Java & object technology into your current software projects.

Up-to-the-Minute, Product Neutral, Immediately Useful – it's like no other event.

This year's technical program has been created by the Tech Chair, Dr. Dwight Deugo, Editor of Java Report. You get the same timely & technically advanced information as found in the well-respected pages of Java Report. It's 5 days of non-stop learning, an unprecedented 75 sessions in all- totaling 162 combined hours of classes over the 5 days. As a software professional you need to explore and exploit the potential of Java 2- as well as its integration with other technologies- such as UML, XML, embedded systems, patterns, and e-commerce. That's why we created a customized event– at the intimate and quietly elegant San Jose Fairmont Hotel. This year's "boutique" event provides you all the skills, tactics, and knowledge you need to build large scale distributed systems with Java.

Nine Tracks on Today's Hottest Topics

- Java Primer
- Programming for the Enterprise (J2EE)
- Programming for the Desktop (J2SE)
- Programming for Devices (J2ME)
- Database integration
- XML Technology
- Object-Oriented Concepts and Design
- Distributed Computing
- Server-Side Programming

Mix and match a customized educational program to fit your needs and gain a competitive advantage. We offer the most product-neutral selection of Java related classes you'll find anywhere.

Is this Conference for you?

SIGS Conference for Java™ Development is designed for members of forward thinking software departments- regardless of industry, business sector, or platform used. Specifically, its for:

- Programmers, developers, engineers
- Software architects & system analysts
- Managers of Intranet/Web Development
- Project Managers/Leaders
- Consultants & Educators

Benefit from Full-Day Tutorials

Immerse yourself in full day (6 hour) tutorials designed to accelerate learning and encourage long-term retention. Choose from 6 tracks on Sunday, October 17, and 5 tracks on Thursday, October 21. These all-encompassing tutorials provide an efficient extension to the 3-day conference (October 18-20) or can be taken independently.

Learn from the best in the business

Get the straight talk (no sales pitches) from those who have pioneered the platform, built the largest Java-based systems, and authored the best selling books. They've already helped thousands hone their programming skills and will do the same for you!

- Bill Venners
- Govind Seshadri
- Andreas Vogel
- Dwight Deugo
- Dave Orchard
- Max Dolgicer
- Scott Ambler
- Mitch Goldstein
- Bertrand Meyer
- Alex Chaffee
- Cay Horstman
- Kent Beck

Night School – Enhance your Program Selection

Why stop when the sun goes down? (no pun intended) Enhance your daily program selection by attending evening classes on Sunday, October 17, Tuesday, October 19 & Wednesday October 20. Choose from 3 or 4 concurrent sessions. One and 1/2 hours in duration. Simply add them to your course selection or just register for night school if you can't get away during the day. (Includes a light dinner.)

'The Road to Java' Getaway Sweepstakes

You may win a **FREE** vacation for 2 (valued at $7,000) to the island of Java. Just come to the exhibit floor and visit the participating vendors! Look for details on our website. Air transportation provided by Continental Airlines. Plus, be a part of the spectacular 'Road to Java' Gala Reception, Wednesday, October 19, where the winner will be announced and featuring Indonesian cuisine, dancing, and music. Sponsored by the US-Indonesian Society.

Call for BOF's

Participating in a "Birds of a Feather" session is a great way to meet, informally, with fellow colleagues and instructors. Brainstorm, share experiences and strategy on a specific area of interest. BOF meetings are scheduled on Monday, October 18 from 6:30-8:30pm. If you'd like to suggest or lead a BOF session, contact Regina Ramazzini at 212-242-7447 ext. 128 or e-mail her at rramazzini@sigs.com before October 4, 1999.

Attention Project Managers – Special Team Discounts!

With 9 tracks and 75 sessions, how can you be certain to get the most out of this event? Cover your bases, send several members of your development team. You'll get the maximum benefit when 3 or more of your team members attend. Each saves $300 off the regular 3-day rate, plus $50 savings on tutorials.

For more information contact SIGS Conferences at: T: (212) 242-7447 • F: (212) 242-7578 • E: conferences@sigs.com • Online: www.javadevcon.com

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries. SIGS Conferences is independent of Sun Microsystems Inc.

Presented by: SIGS CONFERENCES, Java REPORT

Media Sponsors: Application Development Trends, JOOP, C++ REPORT, Windows DEVELOPER'S JOURNAL, C/C++ Users Journal, DEV·X The Development Exchange, Dr. Dobb's, JAVA DEVELOPER'S JOURNAL, SOFTWARE Development, online Java

CORBA components—alive and well

David S. Frankel
dfrankel@gendev.com

CONTRARY TO RUMORS, the OMG CORBA Component Model (CCM) is very much alive. As this article is being written, the latest draft specification has been posted on the OMG Web server[1] (see references 2–5 for other related OMG documents). By the time this article goes to press, it is likely that the process of ratifying the CCM as an OMG standard will be well under way.

CCM Scope and Its Relationship to EJB

There has been a tendency to view CCM simply as a language-neutral version of Enterprise JavaBeans (EJB). Some have even dubbed it "CORBA Beans." While it is true that the CCM architects have gone to great lengths to ensure that CCM plays well with EJB, such characterizations are overly simplistic. CCM has features besides language-neutrality that go considerably beyond EJB.

Container Architecture

CCM and EJB have virtually identical architectural notions of a *container* in which components live. The container is not a visual artifact. Rather, it provides a component with access to a high-level API, to basic services such as transactions, persistence, security, notifications, etc. (see Figure 1). The container API is visible to the component developer but not to the component client.

The EJB container API is a simplified abstraction of the low-level Java services. Because it is a complete abstraction layer, it can actually be implemented by something other than the Java services themselves. In fact, most EJB-based application servers implement this container API via Java versions of the CORBA low-level services, rather than with the Java low-level services (see Figure 2). The application server vendors have made this choice because of the relative maturity of the CORBA services.

However, there is no standard for *how* the CORBA services are used to implement the EJB container API. One of the things that the CCM does is to specify a standard way in which to use the CORBA services to implement the EJB container API. This will be a boon to the industry because it will promote interoperability among application server vendors.

The CCM container API is very similar to the EJB container API; it is very close to being a language-neutral version of the EJB API. It provides a simplified abstraction of the CORBA services, as shown by Figure 3.

Ports

Although the CCM and EJB container architectures are very similar, the CCM model of what a component is is more advanced. CCM components can expose multiple interfaces. Each component definition in IDL declares the interfaces that it *provides* and the interfaces that it *uses*. It also declares events that it *emits* (or *publishes*) and the events that it *consumes*. Provided interfaces are called *facets* and the capability to use an interface is called a *receptacle*. Event emission and publication declarations describe *event* sources and event consumption declarations describe *event sinks*. Collectively, facets, receptacles, event sources, and event sinks are called *ports*.

When an event is published, the container provides the event source with an exclusive event channel; that is, the event source is guaranteed to be the only source using the channel. When an event is emitted, the event channel allotted by the container can also be used by other event sources.

CCM extends CORBA IDL to support declaration of components. The following is an example of a component declaration defining ports.

```
component AcmeARMonthlyUpdate
{
  //ARMonthlyUpdate and Render are interfaces
  //declared elsewhere
  //arUpdate and render are facet identifiers
```

David Frankel is Chief Scientist at Genesis Development Corporation in West Chester, PA. He is a member of the OMG Architecture Board and is co-chair of the OMG's Business Object Initiative Working Group.

Check out our website for info on XML ONE Europe!

XML ONE

The International Users Conference Fall'99

Our Sold-Out Spring Show Returns for Fall '99

Conference: November 8-11 1999 • Exhibit: November 9-10 1999

The Westin Santa Clara – Santa Clara, CA

Stay Ahead Of Your Competition

Today's expert programmers agree – XML is the choice technology for exchanging structured information over the Internet, Intranets and Extranets. **XML One** was created out of the growing need for a developer/programmer/designer oriented industry-wide forum on this emerging technology. So, we've gathered together the top experts in the field and put together 4 action packed days of non-stop learning on the topics you need to know. The fast-paced sessions and jam-packed expo, round out your total XML learning experience.

XML One, strategically located in the heart of Silicon Valley, promises to gather together the most serious developer's from across the country and around the world focused on harnessing the power of XML technology. Our spring **XML One** show sold out in May, with nearly a thousand delegates in attendance.

The "How-to" Conference

You'll want to be there when the XML industry "movers and shakers" unite to define and shape its future in corporate application development. All sessions are filled with straight talk and solid answers on how XML can work for you. Roll up your sleeves side-by-side with the leading experts and work through real world examples and problems. Benefit from personalized attention and question and answer periods.

Technically Advanced and Immediately Useful-Helping You Define Your XML Strategy

You can rely on **XML One** - the "how-to" conference – to keep you and your company current on the state of XML technology. It's the most up-to-date and extensive selection of classes offered anywhere on XML technology. Plus, you'll get tons of fresh ideas and breakthrough innovations from the leading XML tool vendors on the show floor.

Learn from the Leading Experts

Our world-class and unmatched faculty consists of carefully selected experts-from the US, Europe and Canada. They have already taught thousands of developers to be more productive with XML-through their books and lectures. Learn from those practitioners who have successfully built XML systems as well as the most sought-after consultants and book authors.

What You'll Learn:

- definition and understanding of XML basics
- how to use XML-based architecture in Java applications
- how XML fits in to the Web architecture
- a strong understanding of SGML, Xlink, Xpointer, DOM, Extensive Style Sheet language (XSL), XML schemas and DTDs
- how to build a user interface with XML, DHTML and Components
- explore hard-won lessons in XML case-studies- see how to reduce costs & complexity in business applications with XML
- see what advantages in reliability, implementation, speed & cost savings XML provides
- using object databases with XML
- understand how application design is influenced using XML based Java Application Framework
- understand the capabilities and limitations of XFDL & strategies for extending its computational facility using C++ and Java
- see the benefits and solutions using a dynamic XML server
- how to use XML for Windows DNA
- an inside look at emerging new standards & likely outcomes
- the "ins & outs" of the Python programming language
- how to use the freely available Hyperwizard application
- to use the Simple Workflow Access Protocol (SWAP) to transfer structured information encoded in XML
- understand why XML & Message–Oriented Middleware together provide substantial value
- to use XML for inter-component data exchange
- to map your object data to a GUI with XSL
- and much more!

To register:

PH: 212-242-7515

F: 212-242-7578

Email: conferences@sigs.com

URL: www.xmlconference.com

Visit our website at www.xmlconference.com.

We'll give you a free copy of JAVA REPORT just for visiting.

Presented by: SIGS CONFERENCES

Sponsored by: Component STRATEGIES | Java REPORT | XMLInsider | OASIS

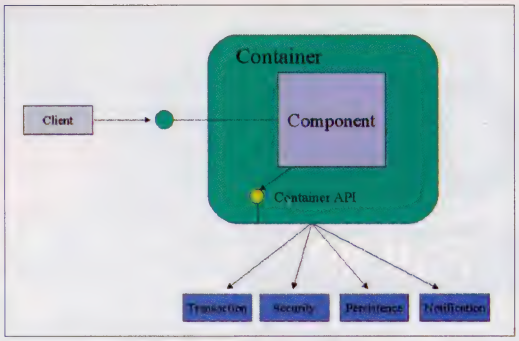


Figure 1. ***General container architecture supported by CCM and EJB.***

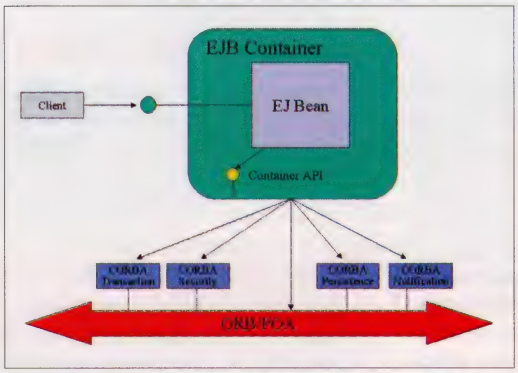


Figure 2. ***Implementing the EJB container API via CORBA services.***

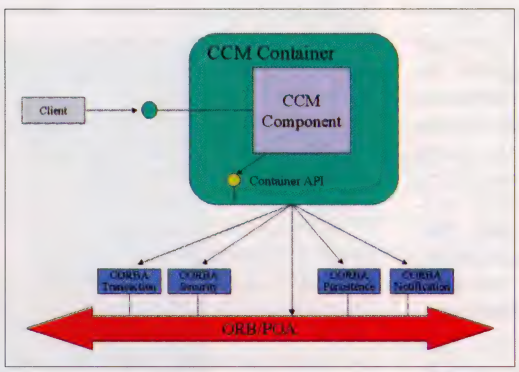


Figure 3. ***The CCM container API simplifies access to CORBA services.***

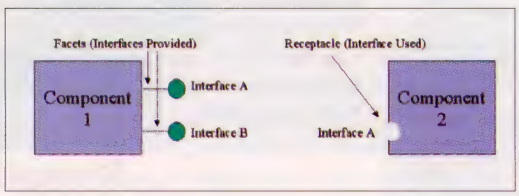


Figure 4. ***A receptacle.***

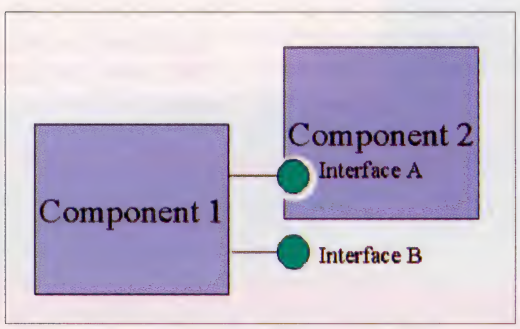


Figure 5. ***Plugging a facet into a receptacle.***

```
    provides ARMonthlyUpdate arUpdate;
    provides Render render;

    //Customer and GeneralLedger are interfaces
    //declared elsewhere
    //customer and gl are receptacle identifiers
    uses Customer customer;
    uses GeneralLedger gl;

    //ARMonthlyUpdateComplete, CollectionsAlert, and
    //OrderInitiated
    //are valuetypes declared elsewhere
    publishes ARMonthlyUpdateComplete
     ARUpdateComplete;
    emits CollectionsAlert collectionsAlert;
    consumes OrderInitiated orderInitiated;
};
```

This example might be used in a situation where an implementation for Acme & Co. of a standardized interface called ARMonthlyUpdate, is to be packaged in a component that also has a notion of how to render an instance of itself to a device and whose rendering behavior can be controlled via a Render interface.

Homes

CCM's concept of a home is very similar to the notion of a home in EJB. CCM defines additional IDL extensions to support declaring homes. A home *manages* a component. The declaration of a home includes the declaration of factories, finders, and primary keys, each of which is optional. More than one home may be declared for a given component type.

The following sample IDL declares a home that manages our example AcmeARMonthlyUpdate component.

```
home AcmeARMonthlyUpdateHome manages
 AcmeARMonthlyUpdate
{
  //CollectionsPolicy is defined elsewhere
  factory CreateARUpdate (
   in CollectionsPolicy policy);
  finder FindARUpdate (
   in CollectionsPolicy policy);
};
```

CCM Components Types

CCM supports four kinds of components:

1. *Entity*—has durable state that must behave as a recoverable, transactional resource, and has a primary key visible to the client. The state is managed by the container. (Homes for entity components typically declare primary keys).
2. *Process*—has durable state managed by the container, but has no client-visible primary key.
3. *Session*—has transient state (also called "conversational" state) that is managed by the component itself.
4. *Service*—is stateless, meaning that it does not carry state from transaction to transaction.

The choice of component type determines a set of POA policies that will be put into force.

Celebrating our 11th Successful Year!

C++ WORLD

The Users Conference and Exhibition

Mark Your Calendar

December 1-3, 1999
Biscayne Bay Marriott, Miami, FL

Created Specially for Advanced Users Building Enterprise-Wide Distributed Systems.

The pages of C++ Report come to life for 3 info-packed days.

4 Tracks, over 30 sessions – presented by the leading writers/thinkers in the language.

Topics include:

- Advanced design patterns
- Large-scale software design in C++
- Advanced template programming
- Avoiding STL "gotchas"
- Applying C++ exceptions effectively
- Optimizing plain & multi-threaded C++ code
- Using COM threading models & ATL
- Mastering software refactoring in C++
- Extending the standard C++ library
- Tips for improving C++ performance
- Mapping objects to relational databases
- Power UML programming techniques

and much more!

For more information:

phone: (212) 242.7515 • fax: (212) 242.7578

e-mail: conferences@sigs.com

Sponsored by: SIGS CONFERENCES, 101 communications

Media sponsors: C++ REPORT, Java REPORT, Windows DEVELOPER'S JOURNAL, Dr. Dobb's JOURNAL, JOOP, C/C++ Users Journal, MSJ Application Development Trends

Visit our web site at: www.cwconference.com for more information

Packaging and Deployment

CCM specifies XML descriptors that contain metadata concerning packaging and deployment. The latest version of EJB also has this feature. However, CCM's packaging and deployment support includes an *assembly descriptor*, which contains metadata describing an assembly of two or more components that have been wired together via their ports. Figure 4 and Figure 5, for example, show an assembly in which a facet is connected to (i.e., plugged into) a receptacle.

CCM specifies two versions of the XML DTDs for the packaging and deployment descriptors. One version is a set of hand-coded DTDs. The other version is a set of XMI-conformant DTDs generated by the MOF–XML mapping applied to metamodels of the descriptor metadata. Which leads us to the subject of the CCM metamodels. (For a discussion of the MOF and XMI, see earlier installments of this column.[6,7,8] The March 1999 installment, in particular, provides a technical overview of the MOF and XMI).

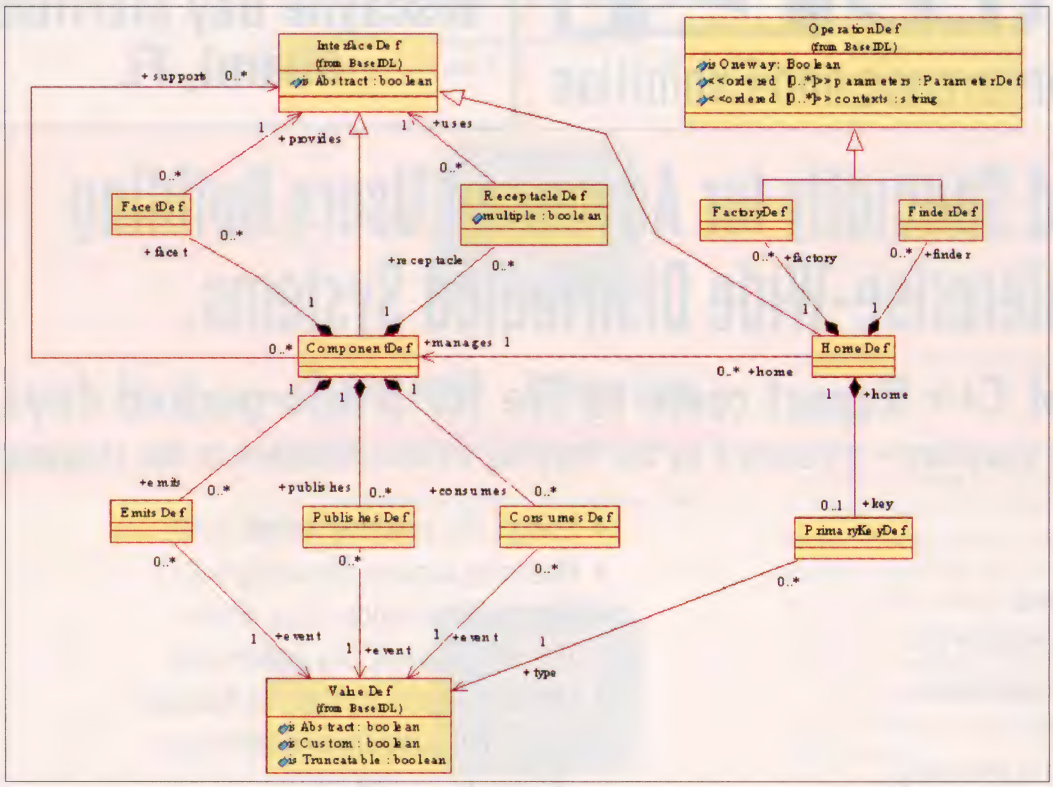


Figure 6. ***Main class diagram for the CCM IDL extensions.***

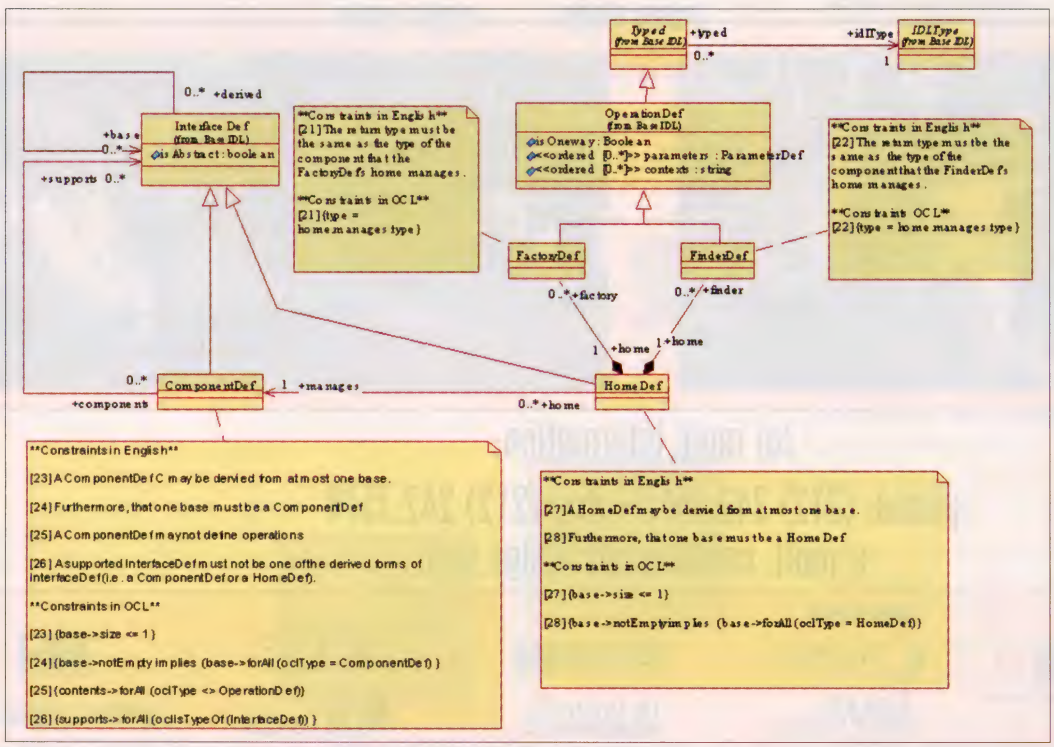

Typed
(from Base IDL)
+typed
+idlType
IDLType
(from Base IDL)
+derived
+base
+supports
InterfaceDef
(from BaseIDL)
is Abstract : boolean
Constraints in English
[21] The return type must be the same as the type of the component that the FactoryDef's home manages.
Constraints in OCL
[21] {type = home.manages.type}
OperationDef
(from BaseIDL)
is Oneway : Boolean
<<ordered [0..*]>> parameters : ParameterDef
<<ordered [0..*]>> contexts : string
Constraints in English
[22] The return type must be the same as the type of the component that the FinderDef's home manages.
Constraints OCL
[22] {type = home.manages.type}
FactoryDef
FinderDef
+factory
+finder
+home
ComponentDef
+manages
+components
HomeDef
Constraints in English
[23] A ComponentDef C may be derived from at most one base.
[24] Furthermore, that one base must be a ComponentDef
[25] A ComponentDef may not define operations
[26] A supported InterfaceDef must not be one of the derived forms of interfaceDef (i.e. a ComponentDef or a HomeDef).
Constraints in OCL
[23] {base->size <= 1}
[24] {base->notEmpty implies (base->forAll (oclType = ComponentDef))}
[25] {contents->forAll (oclType <> OperationDef)}
[26] {supports->forAll (oclIsTypeOf (InterfaceDef))}
Constraints in English
[27] A HomeDef may be derived from at most one base.
[28] Furthermore, that one base must be a HomeDef
Constraints in OCL
[27] {base->size <= 1}
[28] {base->notEmpty implies (base->forAll (oclType = HomeDef))}

Figure 7. ***Some of the formal constraints declared in OCL.***

CCM Metamodels

CCM specifies two MOF-compliant meta models. One is for CCM's extensions to CORBA IDL, and the other is for the packaging and deployment metadata. The formalization of the metamodels is another way in which CCM distinguishes itself from EJB.

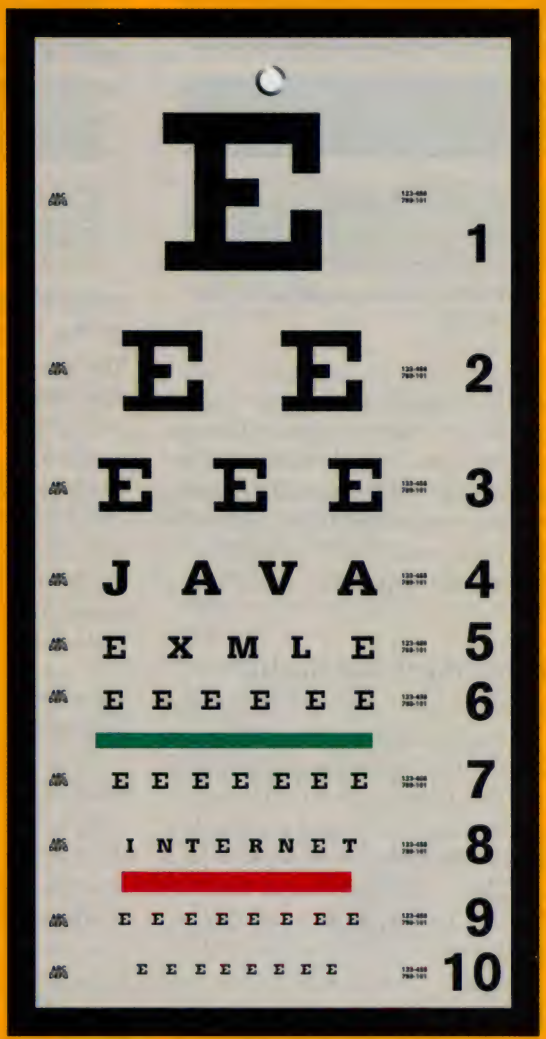
Metamodel of the IDL Extensions

CCM's IDL extensions are by definition based on the pre-CCM CORBA IDL and Interface Repository (IR). However, the OMG specifications for the IR predate the MOF, meaning that a metamodel that extends the IR cannot be MOF-compliant. Therefore, CCM defines a MOF-compliant metamodel for pre-CCM IDL, and from that derives the metamodel for the IDL extensions.

It is important to note that the preexisting CORBA IR is not being deprecated. The CORBA Dynamic Invocation Interface (DII) specification is tailored to the non-MOF IR. Thus, for the fore-

Miller Freeman

Sometimes you need the Big Picture to know where to zoom in. That's why Software Development '99 offers both breadth of technologies and depth of content for building your Internet strategy. SD'99 is the only independent conference that gives you all the options—and lets you decide where to focus your vision. That includes not just Internet technologies like XML and Java, but core languages, enterprise development, and platforms such as Linux and Windows 2000. Best of all, SD'99 lets you mingle with your peers at parties and events, or peer into the future with industry visionaries. Because when it comes to software development, opportunities in the future are closer than they appear.



- JAVA
- C++
- XML
- UML
- INTERNET DEVELOPMENT
- ENTERPRISE DEVELOPMENT
- ENTERPRISE APPLICATION INTEGRATION
- ENTERPRISE JAVABEANS
- CORBA
- COMPONENT-BASED DEVELOPMENT
- MODELING
- COM & COM+
- TESTING
- SECURITY
- LINUX
- USER INTERFACE
- WINDOWS 2000

120 NEW CLASSES WITH OVER 200 TOTAL

Sharpen Your Internet Focus at SD|99

www.sdexpo.com

CONTACT US FOR REGISTRATION INFORMATION
800 441-8826 • sd99@mfi.com

SD|99 software DEVELOPMENT

WASHINGTON CONVENTION CENTER
WASHINGTON D.C.

CONFERENCE NOVEMBER 8—12
EXPO NOVEMBER 9—11

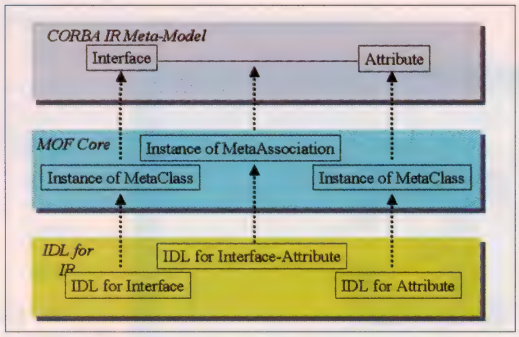


Figure 8. ***The MOF–IDL mapping applied to the CORBA IR metamodel.***

seeable future, these two IR specifications will coexist. They are different representations of the same semantics and therefore, it is fairly straightforward to implement one as a wrapper around the other.

The steps that were followed to produce the CORBA IDL/IR metamodel will be familiar to those who have read the previous installments of this column.

STEP 1: *Declaring the IR Metamodel*

The IR metamodel is expressed in UML notation via Rational Rose class diagrams. The MOF core is sufficiently close to the UML metamodel for it to be possible to use UML notation, albeit with some restrictions, for describing MOF-based metamodels.

The IR metamodel contains two parts, each of which is called a MOF *Package*. The first Package, *BaseIDL*, corresponds to all the features of IDL that predate CCM. The second Package, *ComponentIDL*, corresponds to the new features of IDL described in the submission. Figure 6 shows the main class diagram for the ComponentIDL Package.

An important aspect of the metamodel declaration is the formal specification of constraints. The metamodel expresses constraints using the OMG's *Object Constraint Language* (OCL). The constraints are formalizations of semantics generally expressed via free-form English text in the CORBA Core's specification of the IR (see Figure 7).

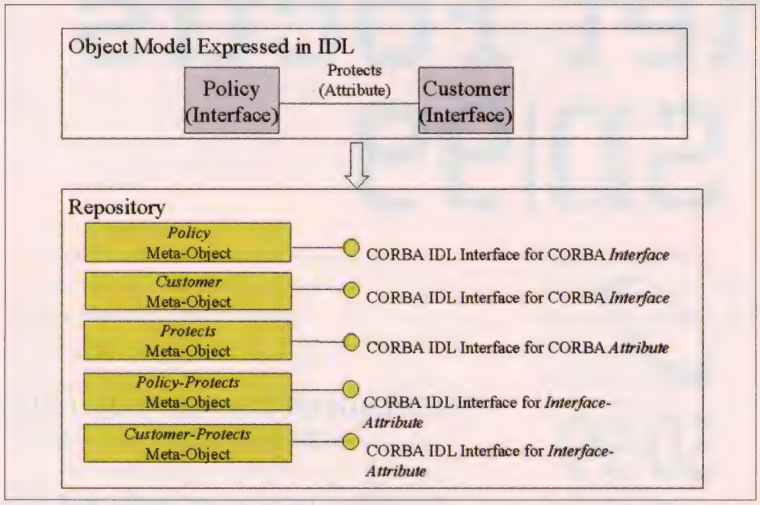


Figure 9. ***Storing a CORBA IDL-based domain model in a MOF-compliant repository.***

STEP 2: *Generating the IDL for the Metamodel*

The IDL for the metamodel was derived by a generator that applied the MOF–IDL mapping to the metamodel declaration (Figure 8). The generated IDL describes the interfaces that the metaobjects will expose in a repository (Figure 9).

STEP 3: *Generating the Implementation of the Generated IDL*

The properties shared by all MOF-compliant metamodels make it possible to generate a great deal of a metamodel's implementation, given a metamodel description. The implementation of the structural portions of the IR metamodel—i.e., of the IDL attributes and operations that result from applying the MOF–IDL mapping to metamodel attributes and to associations—can be generated completely automatically. Implementation of the behavioral portions of the metamodel—i.e., IDL operations that result from applying the MOF–IDL mapping to operations defined on the metamodel elements—must be hand-coded.

A proof of concept prototype of the CORBA IR metamodel jointly developed by the Swiss bank UBS AG, Unisys Corporation, and Genesis Development Corporation was demonstrated at the March OMG meeting. The prototype's implementation of the structural features of the metamodel was, in fact, generated automatically from the metamodel declaration.

STEP 4: *Generating the XMI DTD for the Metamodel*

The XMI DTD for the metamodel was derived by a generator that applied XMI's MOF–XML model to the metamodel declaration (Figure 10). The DTD describes the format used for XML import and export of CORBA IR-based models (Figure 11).

STEP 5: *Building a MOF-Compliant IR Wrapper Around the Non-MOF IR*

This step is actually not typical for MOF metamodels; rather, it is specific to the CORBA IDL/IR metamodel.

The prototype demonstrated at the OMG meeting includes an implementation of the MOF-compliant IR that acts as a wrapper around an implementation of the preexisting CORBA IR standard (Figure 12). It did not present an implementation of the reverse wrapping—i.e., of a non-MOF IR wrapper around an MOF IR—but that is feasible as well. The demonstrated wrapping may be the more compelling, because a number of CORBA product vendors already have functioning implementations of the non-MOF IR.

Packaging and Deployment Metamodel

As mentioned earlier, most of the packaging and

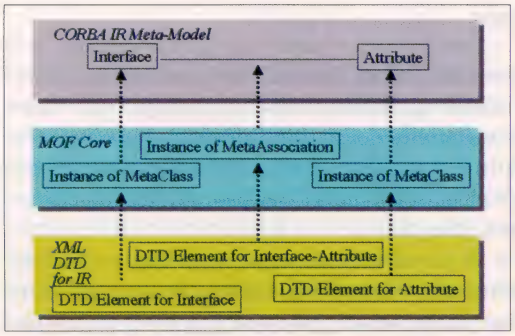


Figure 10. ***The MOF–XML mapping applied to the CORBA IR metamodel.***

deployment metadata are analogous to similar metadata supported by EJB, while the assembly metadata are a notable advance beyond EJB. Space does not allow me to describe the packaging and deployment metamodel. Suffice it to say that it is fully MOF-compliant and thus its IDL and XML DTD were derived via the MOF–IDL and MOF–XML (XMI) mappings, as is the case for all MOF-compliant metamodels.

CCM–EJB Interoperability

I have already discussed the fact that CCM defines a standard way to implement the EJB container API via the CORBA services. It is also important to know that the CCM architects have been careful to ensure that it is very straightforward to implement a CCM component type via EJB, as long as the component type does not use advanced features not supported by EJB, such as multiple interfaces and ports.

The Big Picture—CCM *Plus* A UML Profile For CORBA

A parallel process in the OMG is the RFP for a UML profile for CORBA. There currently is no standard way to use UML to design to the semantics of CORBA IDL. For example, assume I define a UML class to represent a CORBA object type. Is it supposed to represent an interface? Is it a struct? Is it a valuetype? Many people are using UML stereotypes to specify such choices, and a consistent use of a set of such UML stereotypes and tagged values makes it possible to use UML for CORBA IDL-based modeling. A set of such UML extensions essentially comprises a CORBA IR-based UML notation that can be used as a graphical alternative to the text-based IDL.

However, there is no *standardized* set of UML extensions for this purpose. Standardization is necessary in order to promote interoperability among modeling tool vendors. The RFP for a UML profile for CORBA is the process for putting together an OMG standard in this area.

Final submissions to the RFP are due in January, 2000. A UML profile for CORBA covering CCM would, for example, specify UML extensions for modeling ports, homes, and assemblies and for specifying whether a component is an entity, process, session, or service component.

We are thus on the threshold of a very exciting leap in component technology. The combination of CCM and a UML Profile for CORBA that covers CCM will be a powerful marriage of the OMG's two flagship standards: CORBA and UML. ■

References

1. OMG, "CORBA Components Joint Revised Submission," June 3, 1999, document numbers orbos/99-04-16 and orbos/99-04-17.
2. OMG, "Meta Object Facility (MOF) Specification," Oct. 7, 1997, document numbers ad/97-10-02 and ad/97-10-03.
3. OMG, "UML Semantics Version 1.1," Sept. 1, 1997, document number ad/97-08-04.
4. OMG, "Object Constraint Language Specification," Sept. 1, 1997, document number ad/97-08-08.
5. OMG, "XML Metadata Interchange (XMI)," Oct. 20, 1998, document numbers ad/98-10-05 and ad/97-10-06.
6. Frankel, D., "Technology-independent business objects—the concept of a meta-model," *Java Report*, Vol. 3, No. 11, Nov. 1998, pp. 71–78.
7. Frankel, D., "The OMG Meta-Object Facility," *Java Report*, Mar. 1999, Vol. 4, No. 3, pp. 56–62, 71.
8. Frankel, D., "The UML Meta-Model," *Java Report*, June 1999, Vol. 4, No. 6, pp. 90–102.

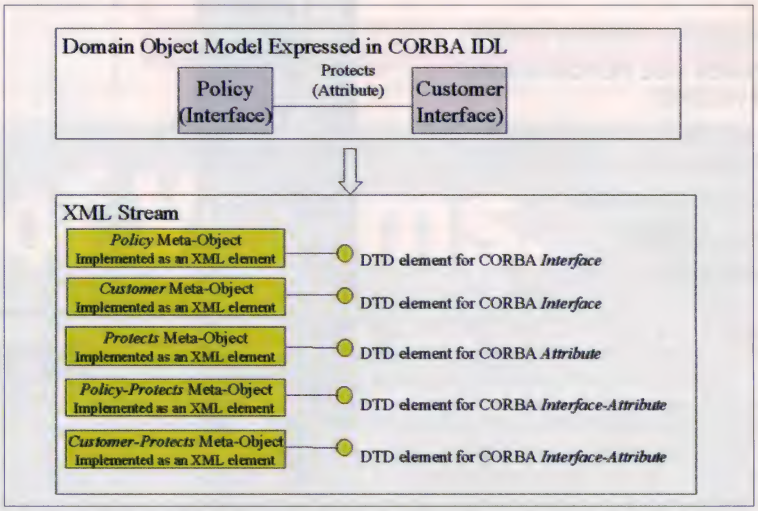

Domain Object Model Expressed in CORBA IDL
Policy (Interface)
Protects (Attribute)
Customer Interface)
XML Stream
Policy Meta-Object Implemented as an XML element
DTD element for CORBA Interface
Customer Meta-Object Implemented as an XML element
DTD element for CORBA Interface
Protects Meta-Object Implemented as an XML element
DTD element for CORBA Attribute
Policy-Protects Meta-Object Implemented as an XML element
DTD element for CORBA Interface-Attribute
Customer-Protects Meta-Object Implemented as an XML element
DTD element for CORBA Interface-Attribute

Figure 11. ***A CORBA IDL-based object model represented in an XMI stream.***

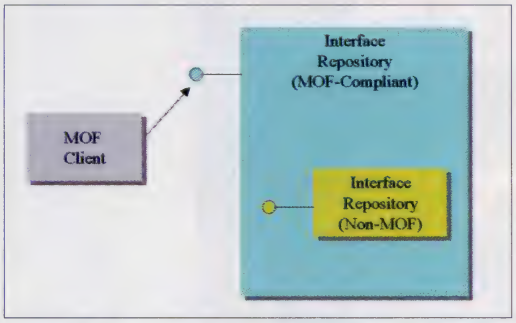


Figure 12. ***Wrapping the Non-MOF IR with the MOF-based IR interfaces.***

Quantity reprints of this article can be purchased by phone:717-399-1900 ext. 139 or by email: sales@rmsreprints.com.

SANCHEZ

JAVA

It's not always easy having one of the hottest products in the banking software market. But it sure is fun! Imagine a place where the entrepreneurial spirit still runs deep. A place where your ideas are recognized and rewarded. Where your career soars!

Working as part of a team, you will be responsible for:

- **Developing projects through all phases of the Software Development Life Cycle**
- **Providing ongoing support/problem resolution, particularly in the framework level and utilities/tools development and Java coding side**
- **Serving as a technical resource for other programmers**
- **Attending information sessions/ training classes**
- **Work closely with clients, including traveling to their sites**

Current Java openings include:

- **SERVER-SIDE PROGRAMMING DEVELOPER**
- **MANAGER - APPLICATION FRAMEWORK DEVELOPMENT**
- **SR. PROGRAMMER**
- **PROGRAMMER**
- **PROGRAMMER/ANALYST**

At Sanchez, we believe the well being of our employees contributes directly to our overall success. We're in the midst of a huge growth phase and we need self motivated people with the energy and enthusiasm to keep driving our entrepreneurial spirit. From casual day every day to our annual turkey derby (you won't believe it until you see it), the key is communication and teamwork.

Enjoy a corporate culture that encourages independence and innovative thinking. We offer very competitive salaries, tuition reimbursement, company stock options, and one of the most comprehensive benefits packages in the industry. For more information and complete job descriptions, visit our website at www.sanchez.com or forward your resume and salary requirements to:

Sanchez Computer Associates, Inc.
Attn: Human Resources/Java
40 Valley Stream Parkway
Malvern, PA 19355
Fax: 610-296-7371
E-mail: jobs@sanchez.com EOE

Work hard. Play hard. Apply at **www.sanchez.com**

SCOTT'S SOLUTIONS *continued from page 80*

Fields() method returns all the fields in our class and its superclasses, but the getDeclaredFields() method gets only those fields that are actually defined in this class. If we're interested in testing to see if all the fields in our class are the same in both objects, we have to take some additional steps as shown in Listing 2.

The loop is required because the getDeclaredFields() method returns only those fields defined in the class. Any fields that are inherited are not returned, even if they are public fields that would ordinarily be returned by the getFields() method.

During the second—and all subsequent—iteration of this loop, one class will be examining the private variables of its parent class. That operation is not normally allowed by

LISTING 1.

The equals() method written using object reflection.

```
public boolean equals(Object o) {
    if (!(o instanceof MyClass))
    return false;
    Class c = getClass();
    try {
        Field[] fields =
                c.getFields();
        for (int i = 0;
            i < fields.length; i++) {
            Field f = fields[i];
            Object old = f.get(this);
            Object new = f.get(o);
            if (!old.equals(new))
                return false;
  }
    } catch (Exception e) {
       return false;
    }
    return true;
}
```

LISTING 2.

Testing to see if all fields in our class are the same in both objects.

```
public boolean equals(Object o) {
    try {
        Class c = getClass();
        while (c != null) {
            Field[] f =
                c.getDeclaredFields();
            for (int i = 0;
                    i < f.length; i++) {
                f[i].setAccessible(true);
                Object old = f[i].get(this);
                Object new = f[i].get(o);
                if (!old.equals(new))
                    return false;
            }
            c = c.getSuperclass();
        }
    } catch (Exception e) {
        return false;
    }
    return true;
}
```

Java, and it is only allowed in this example if we have called the setAccessible() method on the field before obtaining the field's value. The setAccessible() method is only available in the Java 2 platform, so this technique will not work in previous releases of Java.

Calling the setAccessible() method allows us to bypass the normal security checks that the get() method would perform, which is what allows us to obtain the value of the private fields in our parent's class. However, the setAccessible() method itself can only be called if the currently running code has the appropriate permission: a java.lang.reflect.ReflectPermission with the name "suppressAccessChecks." You wouldn't necessarily expect this permission to be set (nor should you agree to set it yourself without strict assurances that the code that grants such permission is really to be trusted).

If you can't use the setAccessible() method, then you can still use this technique for the private objects that are locally defined in the class; you must merely remove the loop that tests each superclass.

This last example has many potential uses. For example, you could define an XMLSupport class that provided a toXML() method that looks just like our equals() method, except rather than testing the equality of each field, it could convert each field into an XML string and paste the whole thing together. You can use this technique to identify and deal with transient fields that you encounter in object serialization. And so on. A little reflection will lead you to your own solutions and to more powerful programs. ■

Salaries $150,000+

Relocation Not Required

AvantSoft, Inc.

AvantSoft provides exceptional quality training in state-of-the-art areas of Java, EJB and XML.

Java Instructor

Responsibilities include teaching and developing courses, technology exploration and consulting. Applicants must have excellent industry teaching and development experience (3+ years) in **Java, J2EE, EJB, XML, Swing, Security and CORBA**. Travel required.

Business Development Manager

Applicants must have 3+ years of experience with a high-tech training and consulting company, including the establishment of customer accounts and consulting projects, marketing, recruitment and market analysis.

Please send resume to:
Fax: 408-530-8703
Email: jobs@avantsoft.com

www.avantsoft.com

A Can of Worms.

Casting for the best Java jobs?

Cut bait with the online bulletin boards and resume databases. Finding a great Java job is easy with Career Central. We reel in the jobs that match what you want. Best of all, it's free.

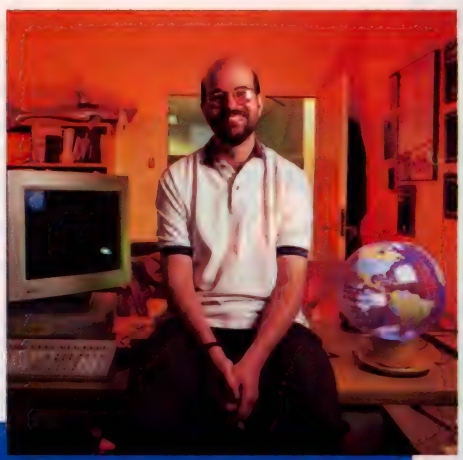
career central
for java developers

Sign up at www.careercentral.com/java

Using the Reflection API

WE'RE GOING TO explore some novel ways of using the reflection API. I've had a few problems lately that had the same answer as we are about to explore; such is the flexibility of the technique that we're examining. The starting point for our exploration is a solution first shown to me by Mike D'Orazio and Ray Washburn at AT&T; I immediately grabbed it, thus promoting code reuse.

The basic question that we were interested in exploring is when are two objects the same? This is the question that is posited by the equals() method of the Object class, which allows us to test if any one object is the same as any other object. But what does it really mean to be the same?



Scott Oaks
soaks@sigs.com

The answer to that, of course, varies depending on the objects that we are interested in. The default answer—provided by the implementation of the equals() method of the Object class—is that two distinct objects are never really the same: The equals() method only returns true if the object references in question point to the same memory location. That is, object one will be equal to object two only if we follow this sequence:

```
Object one = new Object();
Object two = one;
```

What about String objects? Two String objects are considered the same if they represent the same sequence of characters, even if they are distinct objects. Hence, we can do the following:

```
String one = new String("Hello");
String two = new String(one);
```

In this case, one.equals(two) will return true, even though one and two are different objects.

In general, it is up to an object to determine its own rules of sameness: Strings are equal if they represent the same sequence of characters; URLs are equal if they have the same protocol, port, and file parts; Integers are equal if they wrap the same number; and JButtons are equal only when we're talking about the same reference.

What if we wanted something a little more generic? What if we wanted to say that two objects were the same if each of their data elements—that is, each of their fields—were the same. We could program that explicitly as follows:

```
public class MyClass {
        public String one, two;
        public boolean equals(Object o){
            if (!(o instanceof MyClass))
                    return false;
                MyClass o2 = (MyClass) o;
                return
                    o2.one.equals(one) &&
                    o2.two.equals(two);
            }
        }
```

The problem with this approach is that if we add (or remove) data objects to the definition of MyClass, we have to remember to update the equals() method. Using object reflection instead, we can write the equals() method, as shown in Listing 1.

There are some caveats with this code: The first is that when we test the embedded objects for equality, we're again subject to the definition of their equals() method. So in our example, where the instance variables are Strings, that will work just fine, but if there are embedded data elements of other types, we might not get exactly the results that we want.

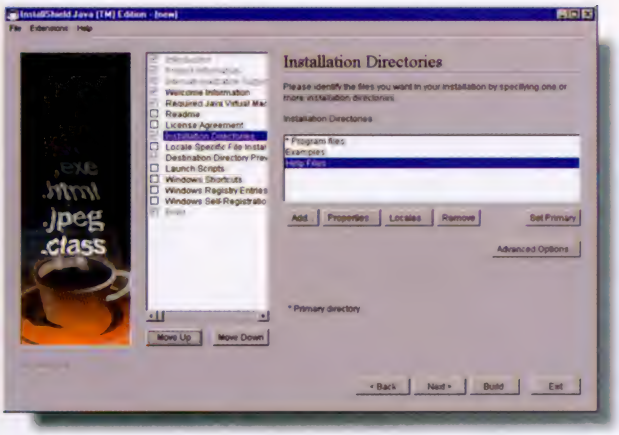
The second issue is which data elements we're interested in testing. In the above example, we used the getFields() method to determine this, which means that we'll test all public fields that are in this class as well as all public fields in each of this class' superclasses. If we are interested in private, protected, or default-permissioned fields, then we'd need to use the getDeclaredFields() method instead.

But there's a subtle difference here: The get-

continued on page 78

Scott Oaks is a Systems Engineer for Sun Microsystems, where he focuses on practical applications of Java technology. He is the co-author, with Henry Wong, of *Java Threads*.

InstallShield® JavaTM Edition Version 2.5

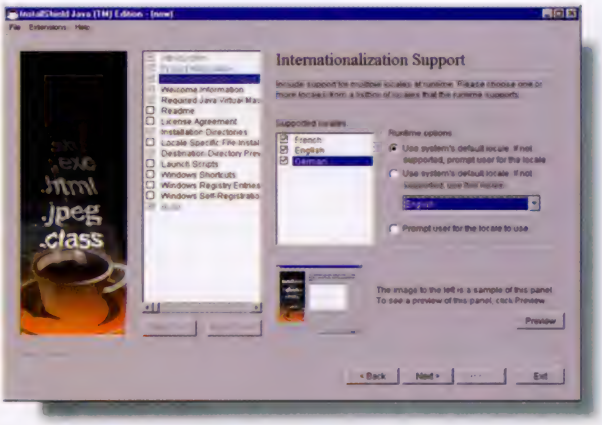


Multiple source and target directory support.

Through the InstallShield Java Builder, you can deploy files to multiple directories on a target system. As you build your setup, locate files in a clear, customized set of directories that will remain correctly distributed on your customer's machine after installation. No more cluttered directories!

Internationalization support.

You can display your custom extensions in different languages. InstallShield Java Edition now includes language resources for all standard end-user dialogs in French, German, Italian, Japanese, and Spanish. Use the **new internationalization APIs** to update your custom extensions, and your setup is ready for the world!


Internationalization Support
Supported locales
French
English
German
Runtime options
Use system's default locale. If not supported, prompt user for the locale
Use system's default locale. If not supported, use this locale
Prompt user for the locale to use
The image to the left is a sample of this panel.
To see a preview of this panel, click Preview.
Preview
< Back
Next >
Exit

NEW VERSION!

InstallShield® Software Corporation

The total solution for Java™ deployment.

InstallShield® Java Edition provides developers with a visual tool designed to create a single Java installation that's consistent in look, feel, and functionality across multiple platforms. New InstallShield Extensions for the Java platform bring the power of Java to installation building, allowing you to plug your own Java-based extensions into your installation. And the InstallShield Installation Applet lets you create a seamless cross-platform launch site for your installation. Best of all, InstallShield Java Edition is authored by InstallShield Software Corporation, the worldwide leader in installation and distribution technology.

Exclusive features in InstallShield Java Edition 2.5

- True cross-platform installations—no need for multiple installations for individual platforms
- InstallShield Extensions for the Java platform give you the power of Java to build your installations—plug in your own Java-based extensions or use one of the pre-built examples
- InstallShield Installation Applet lets your users specify or get the appropriate Virtual Machine (VM), download, and install your Java application by clicking a Web link
- Self-extracting installations allow users to launch installation with point-and-click ease
- Command-line access allows you to add installations to your batch file building process
- Specify multiple source and multiple target directories for your installations
- Create installations in English, French, German, Italian, Japanese, and Spanish

©1999 InstallShield Software Corporation. All rights reserved. InstallShield is a registered trademark and service mark of InstallShield Corporation. "It All Starts Here" is a service mark and a registered trademark of InstallShield Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and/or other countries. Part No. 181-20009 0199

It All Starts Here®

900 National Parkway, Suite 125
Schaumburg, IL 60173-5108 USA
Sales: +1-847-240-0618
Fax: +1-847-619-0788
Toll-free: +1-800-440-8149
sales@installshield.com
www.installshield.com

InstallShield® SOFTWARE CORPORATION

www.installshield.com/java

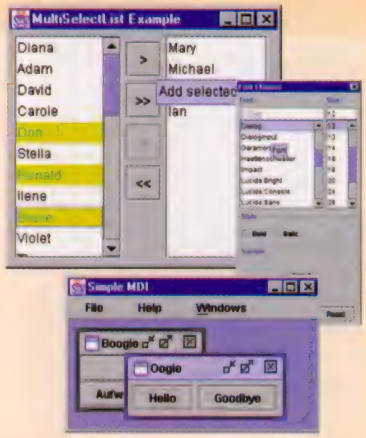
Put some power into your Java™ apps

JClass SwingSuite: enhance the power of Swing

JClass SwingSuite is a premium collection of components and Beans that substantially extend and enhance the core Swing toolkit. Improve the look and feel of your Java applications easily and quickly. For a free trial, visit:

www.klgroup.com/swingsuite

- Multi-Document Interface (MDI) component
- Powerful wizards and layout managers
- Multi-column outliner
- Tree explorer
- Spin and list boxes
- Font and date choosers
- and much more!



JClass JarMaster: the fastest way to optimize JAR files before deployment

A powerful Java utility, JClass JarMaster selects only those files you really need to build JAR or ZIP files – no more, no less. By selecting and verifying all files contained in your JARs, JarMaster creates the smallest possible archive, while ensuring that nothing critical is left out. Optimize your JARs today. For a free trial, visit:

www.klgroup.com/jarmaster

- Rapid GUI or command line JAR building
- Easy-to-use JAR-building wizards
- Advanced interface for adding, deleting, or transferring classes
- Easy drill-down access to determine dependencies
- Edit and explore multiple JARs simultaneously
- Bean manifest editor for customizing JAR manifest

Editors' Choice JavaWorld 1998

JDJ Editor's Choice Award

KL GROUP
Software Development Productivity

KL Group Inc. Toll-Free: 1-888-361-3264 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

© Copyright 1999 by KL Group Inc. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. JClass, JClass SwingSuite and JClass JarMaster are trademarks of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. JR011

For Faster Service & More Information:

FAX
615.370.4845

E-MAIL
subscriptions@sigs.com

ON-LINE
www.javareport.com

CUSTOMER SERVICE
1.800.361.1279

SIGS PUBLICATIONS

Java REPORT

SUBSCRIPTION SAVINGS COUPON

YES!

Enter my subscription at the term below. This is a risk-free offer. If I'm not completely satisfied I may cancel and get a prompt refund for all unmailed issues.

- ❑ 1 year—$39 (12 issues)
- ❑ **2 years—$62** (24 issues) save 20%

B90703

❑ **Check enclosed** (payable to ***Java™ Report***)

Charge my ❑ Visa ❑ Mastercard ❑ American Express ❑ Bill me

Card # Exp. date

Signature

Name

Title

Company

Address

City

State/Province Zip/Postal Code Country

Phone Fax

E-mail *We reserve the right to rent our e-mail list*

❑ Check here to receive only SIGS' promotions via e-mail

Make checks payable to ***Java™ Report***. Canadian and Mexican orders must add $25 per year. Checks must be in US dollars and drawn on a US bank. Canadian GST included (#R895884229)

Java and Java-based marks and trademarks are registered trademarks of Sun Microsystems, Inc. in the United States and other countries. ***Java™ Report*** is independent of Sun Microsystems, Inc.

Place
Postage
Here

PO Box 5050
Brentwood, TN 37024-9713

For Faster Service & More Information:

FAX
615.370.4845

E-MAIL
subscriptions@sigs.com

ON-LINE
www.javareport.com

CUSTOMER SERVICE
1.800.361.1279

B90703

For Faster Service & More Information:

FAX
615.370.4845

E-MAIL
subscriptions@sigs.com

ON-LINE
www.sigs.com

CUSTOMER SERVICE
1.800.361.1279

SIGS PUBLICATIONS

YES!

Enter my subscription to the following magazine(s) at the term below. This is a risk-free offer. If I'm not completely satisfied I may cancel and get a prompt refund for all unmailed issues.

C++ Report

- ❑ $79 One Year (10 issues)
- ❑ $126 Two Years (20 issues) save 20%

Java™ Report

- ❑ $39 One Year (12 issues)
- ❑ $62 Two Years (24 issues) save 20%

JOOP

- ❑ $79 One Year (9 issues)
- ❑ $126 Two Years (18 issues) save 20%

Deduct 10% for orders of two or more magazines.

❑ **Check enclosed** (payable to SIGS)

Charge my ❑ Visa ❑ Mastercard ❑ American Express ❑ Bill me

Card # _____ Exp. date _____

Signature _____

Name _____

Title _____

Company _____

Address _____

City _____

State/Province _____ Zip/Postal Code _____ Country _____

Phone _____ Fax _____

E-mail _____

We reserve the right to rent our e-mail list.

❑ Check here to receive only SIGS' promotions via e-mail.

Make checks payable to SIGS Publications. Canadian and Mexican orders add $25 per year per title. Checks must be in US dollars and drawn on a US bank. Canadian GST included (#R895884229)

PLACE
POSTAGE
HERE

PO Box 5050
Brentwood, TN 37024-9713

For Faster Service & More Information:

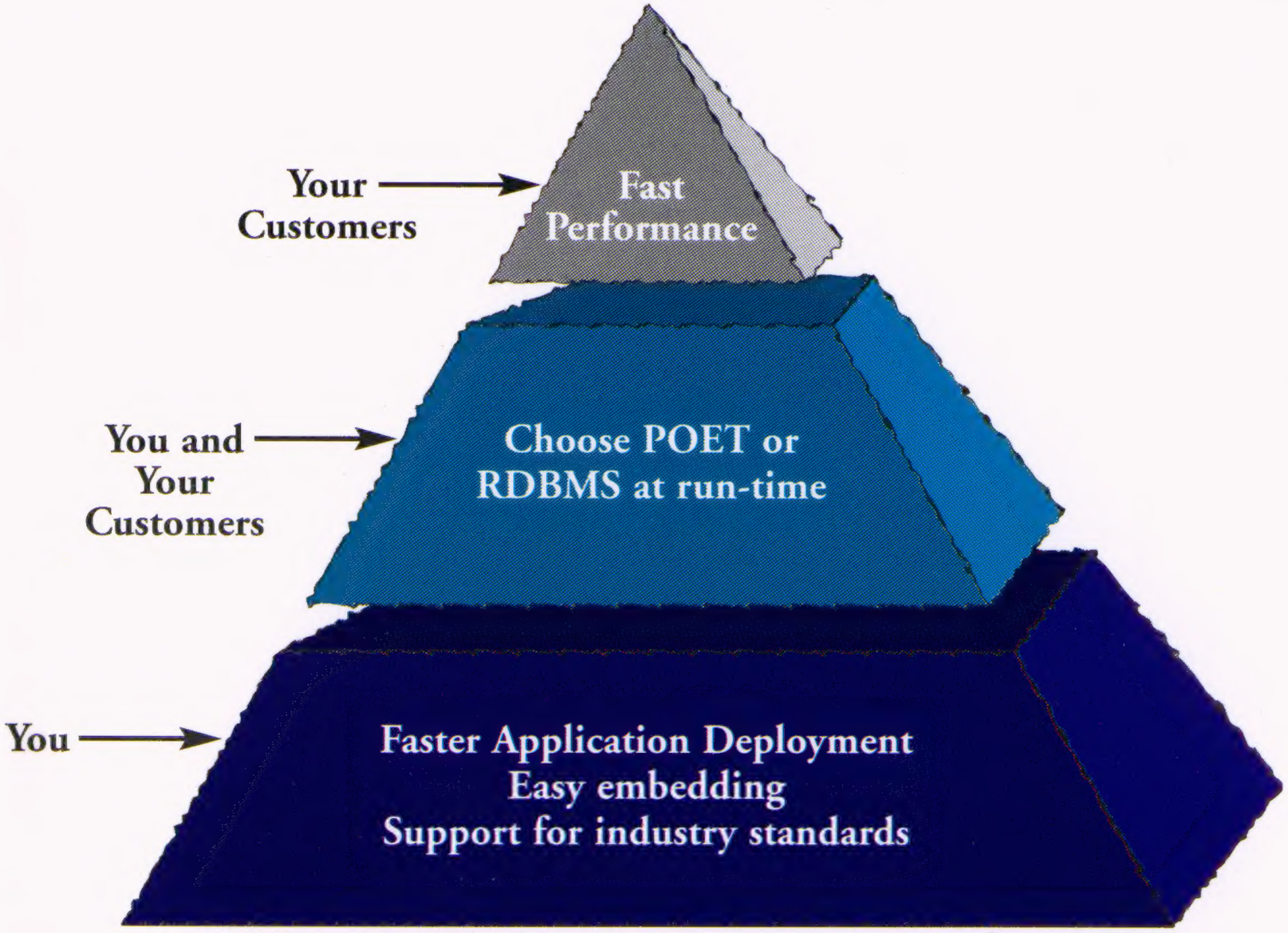
FAX
615.370.4845

E-MAIL
subscriptions@sigs.com

ON-LINE
www.sigs.com/html/publications.shtml

CUSTOMER SERVICE
1.800.361.1279

Who Benefits from POET FastObject™ Technology?



POET Object Server Suite

The Essential Database for Java and C++ Objects

POET

For more information visit our website at **http://www.poet.com**

Need Fast Storage of Java and C++ Objects?

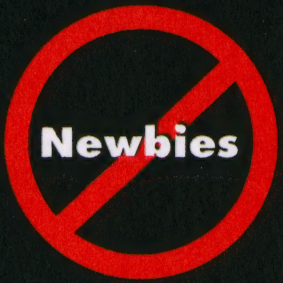
The fastest way to store Java and C++ objects!

See back for details

POET
Object Server Suite
featuring
POET FastObject™ Technology

FREE
TRIAL DOWNLOAD
www.poet.com

POET



You're Not a Newbie—
Read JOOP
1 Year (9 issues) $79

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

❑ Bill me later ❑ Payment enclosed

Canadian and Mexican price is $104. Checks must be in U.S. dollars and drawn on a U.S. bank. Canadian GST included (#R895884229)

U90501

JOOP
The Global Authority on Object Oriented Development
The Journal of Object Oriented Programming

SIGS
Phone: 212.242.7447
Fax: 212.242.7574

www.joopmag.com

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 2 BRENTWOOD, TN

POSTAGE WILL BE PAID BY ADDRESSEE

SIGS PUBLICATIONS INC
PO BOX 5049
BRENTWOOD TN 37024-9713